GitHub Username: **sanjnair**

Mélange – Indian classical music learning platform

Description

This app enables a student of Indian classical music to learn music from a classically trained music teacher (Guru). Traditional form of south Indian classical music learning is very structured. Lessons start with introducing the students to basic notes (*swaras*) in a particular scale (*Raga*). Following that students are given progressively complex lessons. Music teacher also provides several resources such as *music scale* information, *pattern* information, *lesson recordings* (typically audio), *related work* (audio/video/text resources) etc.

Intended User

This app is intended for students and teachers of Indian classical music. This app will connect the students with teachers and provide number of features to enhance the experience of quality learning.

Features

- **Main** Features (planned for version 1 – for Udacity course)
  - Enables a student to sign into the course (limits one course for version 1).
  - Enables students to view the course lessons.
  - Enables students to download the course content and view (or play) different types of content on the mobile device.

        - Different types of content can be text, audio and video contents
    - Keeps track of lesson progress

- **Optional Features** (planned for future releases – not for Udacity course)
    - Enables a user to search for a music teacher and initiate the registration process
    - Enables student to upload the versions of practice results (such as audio) for the teacher to check
    - Enables teacher to check the student uploaded content and provide feedback.
    - Enable push notifications for the student and teacher.
    - Manage payment details (such as registration fee, subscription fee, lesson fee etc.)

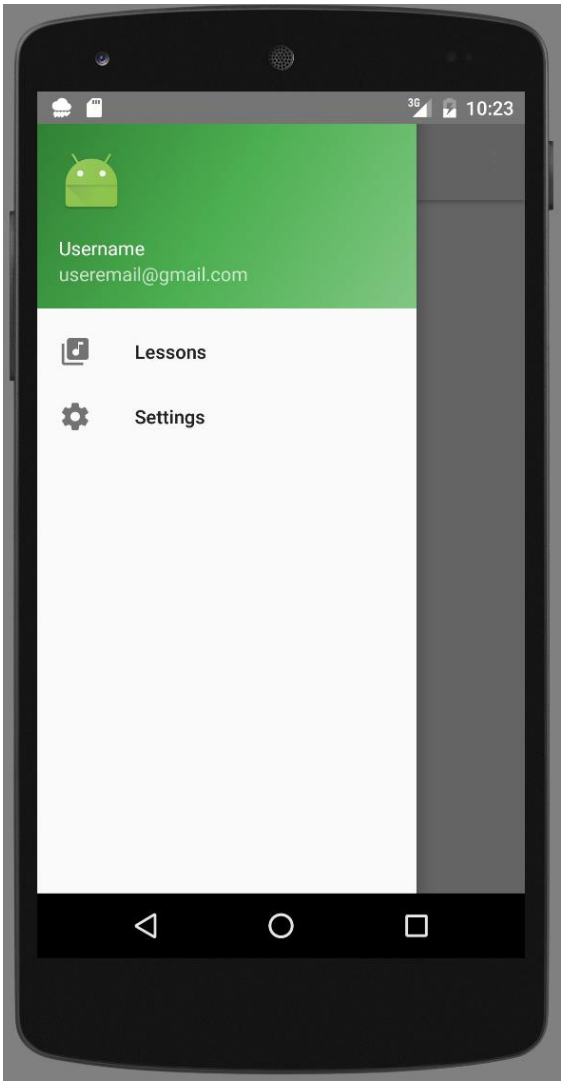
User Interface Mocks

1. Start / Login Screen

This screen is the first screen presented to the user. It is required that user login to the system before he/she can start using the system. For the first phase, we'll use Google's login capabilities to authenticate the system. On subsequent release, we'll consider adding facebook support as well as our own login database.

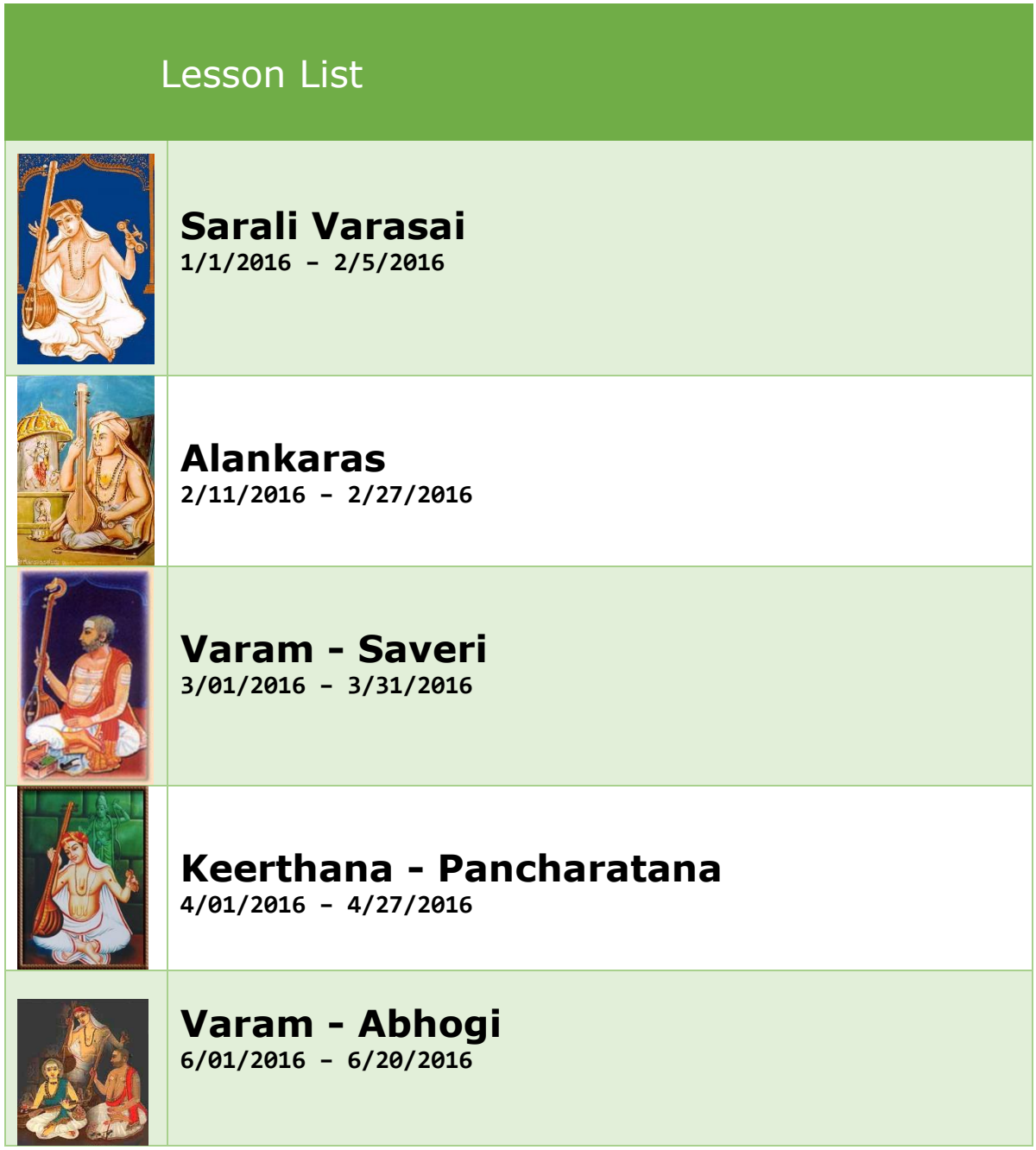
2. Navigation Menu

This page mainly used for navigating different parts of the app. Currently, only two features are supported as shown below.



3. Lessons list Screen

This screen provides a list of lessons sorted by the lesson date (or order or teaching). Student can select a lesson to view more details. This page will be of Master-detail style so that user can view the details in the same page when viewing on large screen devices (such as tablets).



4. Lessons Detail Screen

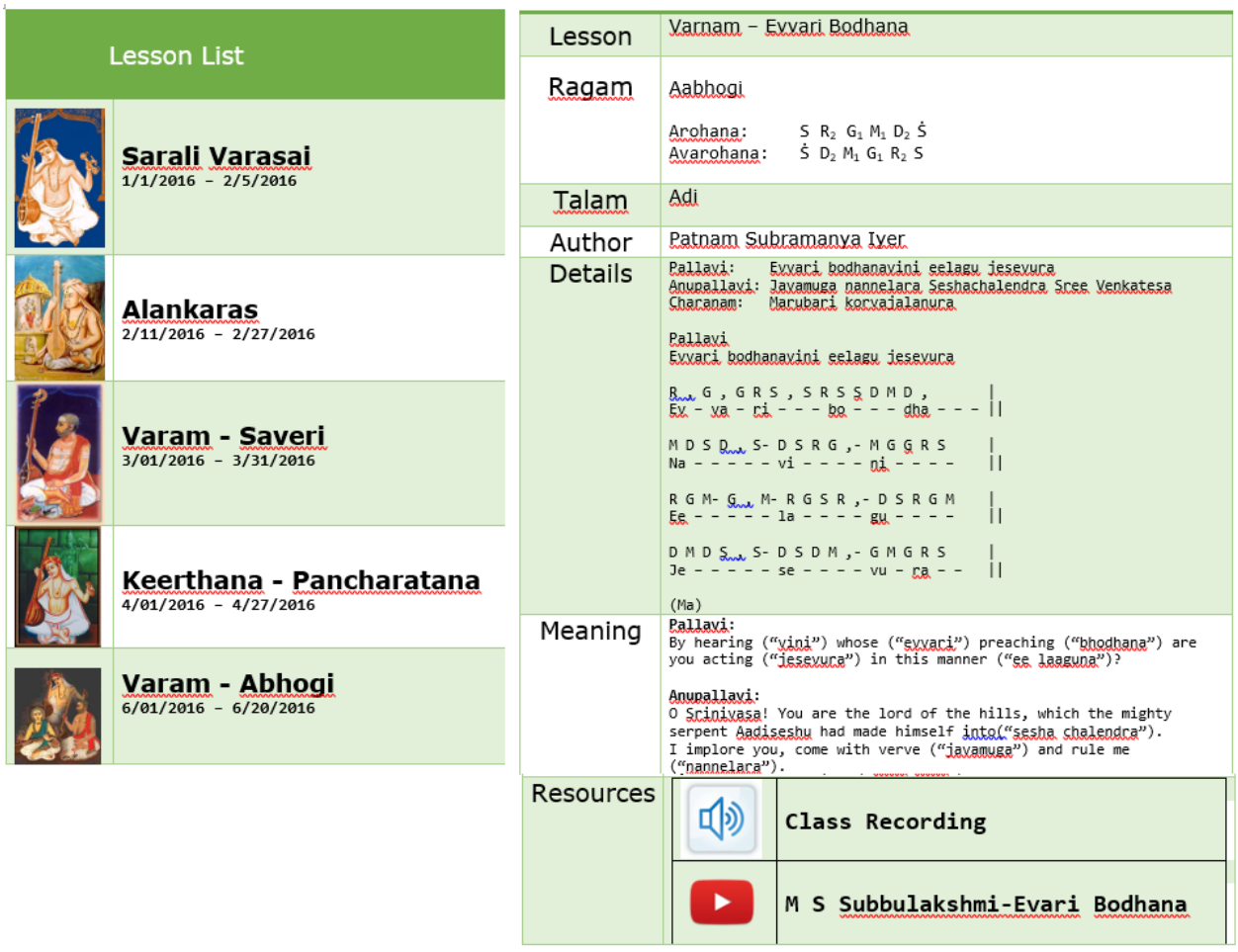
This screen provides lesson details. Details such as raga (scale), thala (rhythm), musical notation as well as additional resources.

← Lesson Details	
Lesson	Varnam – Evvari Bodhana
Ragam	Aabhogi Arohana: S R_2 G_1 M_1 D_2 Ṡ Avarohana: Ṡ D_2 M_1 G_1 R_2 S
Talam	Adi
Author	Patnam Subramanya Iyer
Details	Pallavi: Evvari bodhanavini eelagu jesevura Anupallavi: Javamuga nannelara Seshachalendra Sree Venkatesa Charanam: Marubari korvajalanura Pallavi Evvari bodhanavini eelagu jesevura R , G , G R S , S R S S D M D , | Ev – va – ri – – – bo – – – dha – – – || M D S D , S- D S R G ,- M G G R S | Na – – – – – vi – – – – ni – – – – || R G M- G , M- R G S R ,- D S R G M | Ee – – – – – la – – – – gu – – – – || D M D S , S- D S D M ,- G M G R S | Je – – – – – se – – – – vu – ra – – || (Ma)
Meaning	**Pallavi:** By hearing (“vini”) whose (“evvari”) preaching (“bhodhana”) are you acting (“jesevura”) in this manner (“ee laaguna”)? **Anupallavi:** O Srinivasa! You are the lord of the hills, which the mighty serpent Aadiseshu had made himself into(“sesha chalendra”). I implore you, come with verve (“javamuga”) and rule me (“nannelara”).

	Charanam: I cannot endure (“orva jalanura”) the suffering caused by the deeds of cupid (“maru bari”).
Resources	 **Class Recording**  **M S Subbulakshmi-Evari Bodhana**

Clicking on Resources will open up appropriate third party applications such as music player / youtube player (or browser).

Master Detail View



5. Lessons Widget

This app will also provide a widget that can be pinned to the home screen. This widget shows how many lessons completed and how many lessons are not completed. Please note that for version 1, this information may be stored locally. Also, for version 1, user will have to mark a lesson as complete in the lesson detail screen (a checkbox will be provided).

Key Considerations

- App will use google cloud service to store and fetch data. Typically, teacher is responsible for pushing the data to the cloud service and the app will download the data from the cloud service.

- For this project, we'll not provide support for pushing the data to cloud. Instead, the data will be populated in the cloud database by other means. We'll use Google Cloud Endpoint service for the cloud support.

- App will use Sync adapters (to sync data from cloud), loaders and content providers to load and fetch data.

- App will use a local database (sqllite or realm) to store the data locally. App GUI will this database to fetch the data and display it.

- For the initial version of the app, app will not push data to the cloud.

- Data flow looks like this:
    - App uses sync adapter to fetch data from cloud and store it in local database and notify the UI load the data from local database
    - UI portion of the app will load the data from local database and display the data to the user.
- UX corner cases
    - From the detail page, user will be returned to the lesson list page when the back button is clicked.
    - App will use image loading libraries such as Picasso or Glide to handle loading and caching of images.

Next Steps: Required Tasks

Task 1: Project Setup

- **Cloud setup**
  - Setup API backend using Google cloud endpoint services
  - Setup cloud object storage using Google cloud storage
  - Setup messaging using GCS (Google Cloud Messaging) or FCS (Firebase Cloud Messaging)
  - Design and create database schema for cloud
- **App setup**
  - Design and create database schema for local storage
  - Select libraries for local database (sql lite, realm), and image loader (Picasso, Glide)

Task 2: Implement UI for Each Activity and Fragment

- **Login Activity**
  - Build UI for Login activity
  - Use Google's authentication API to authenticate user
  - After successful login, go to main activity

- **Lesson list Activity**
  - This is the main activity of the application.
  - Build UI for this activity
  - Load data from local database to populate the UI as per the spec.
  - If no data available, display appropriate message.
  - Provide support for navigating to lesson detail activity after selecting a lesson from the list
- **Lesson detail Activity**
  - Build UI for this activity
  - Load data from local database to populate the UI based on the lesson selected.
  - If no data available, display appropriate message.
  - Provide support for navigating back to the lesson list activity upon selecting the back button.
- **Navigation Menu**
  - Build UI for this menu
  - Provide support for going to lesson list activiy

Task 3: Implement data storage, data fetch/sync

- Implement sync adapter to fetch/sync data from the cloud and store it in local database
- Implement Data loaders to get data from local database for populating the GUI
- Implement Google cloud endpoint API to communicate with cloud storage