How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: “**Capstone_Stage1**”
3. Replace the text in green

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”

GitHub Username: vivekkiran

CodeMozo - Code Calendar

Description

This app is for everyone who has a passion for competitive programming, coding challenges and hackathons. This app lets you keep track of all upcoming, live and past contests on various websites which includes platforms like Codechef, HackerRank, HackerEarth, Codeforces, Topcoder and others. Users can choose to set up reminders for upcoming contests, and get notified before the contest starts. Contests can be added to their calendars directly from the app.

Intended User

This app is for students, programmers, professionals across the globe who love to take part in various online coding competitions and sharpen their skills.

Features

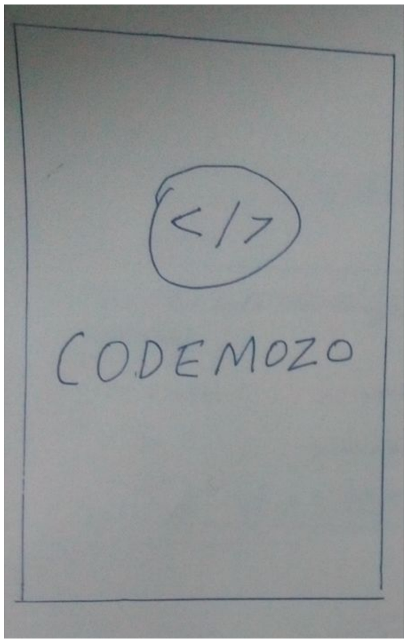
List the main features of your app. For example:

- App shows all live, upcoming and past contests .
- Contests can be categorized past, live and upcoming contests.
- Users can select his preferred websites to see contests from those websites only.
- Feature to show new contest notification.
- Feature to add reminders for a contest .
- Contests can be added directly to the calendar.

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

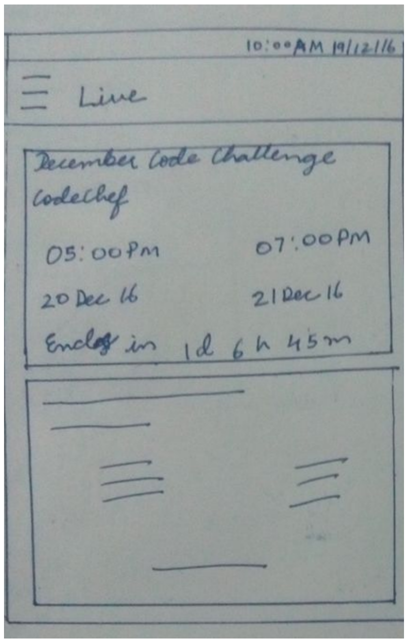
Screen 1: Splash Screen



Splash screen is shown when app is cold started. Screen contains app name, logo, and one geeky one liner.

Behind the scene sync adapter fetches latest data from REST apis and sync it with local sqlite db using content providers.

Screen 2: Live Contests

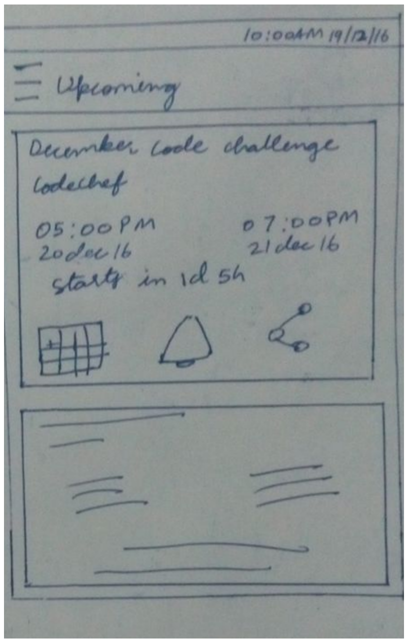


This screen contains list of contests that are currently live. Screen contains a listview, with each item being a cardview with linear layouts containing other widgets.

Each list item shows contest name, website, its logo, and start & end time.

Clicking on any contest takes user to contest detail page.

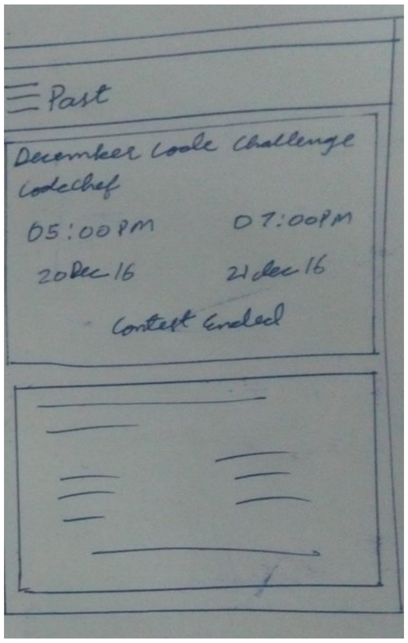
Screen 3: Upcoming Contests



This screen contains list of upcoming contests. User can add these contests to their calendars, add reminders for them, also share them.

Clicking on any contest takes user to contest detail page.

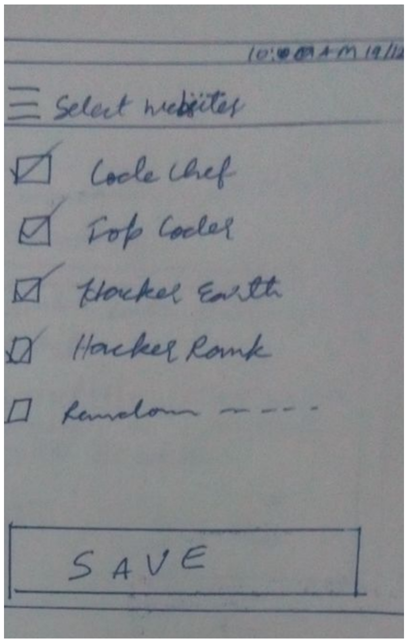
Screen 4: Past Contests



This screen contains list of past contests of last 7 days. We delete data older than last 7 days to keep our local db light in weight.

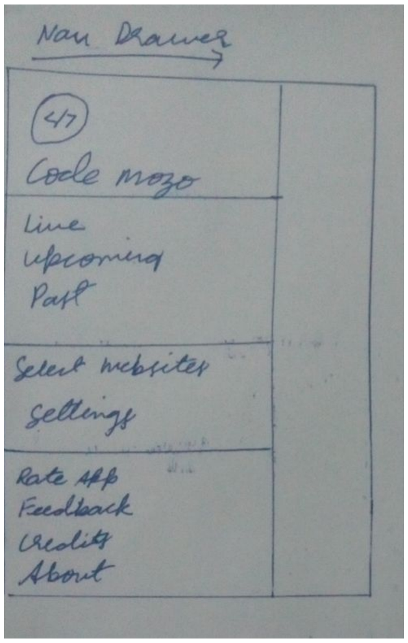
Clicking on any contest takes user to contest detail page.

Screen 5: Select Websites



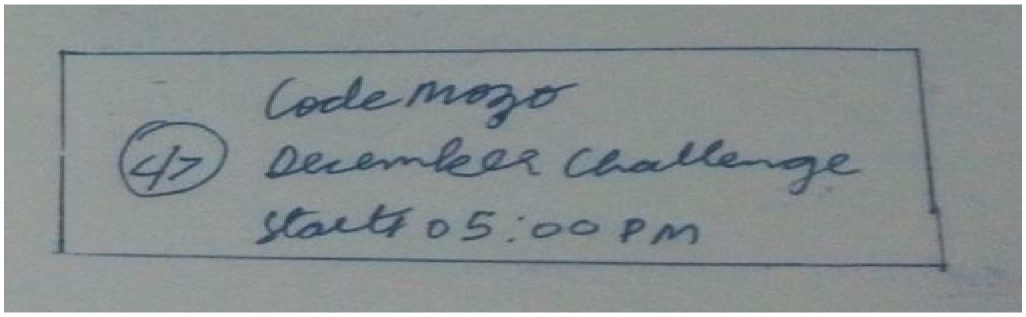
Users can select their favorite websites from which they want to see the contests.
User is presented with a listview, each listview item has checkbox, logo and website name.

Screen 6: Left Navigation Drawer



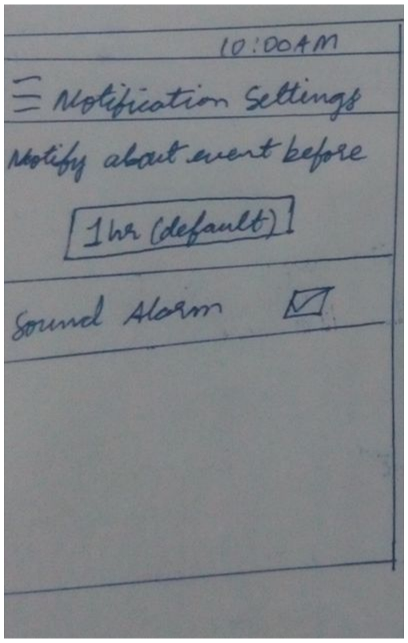
Left Navigation drawer that is opened when user clicks on top left hamburger icon.
This drawer contains various options for showing contests, settings, and feedback related options.

Screen 7: Sample Notification



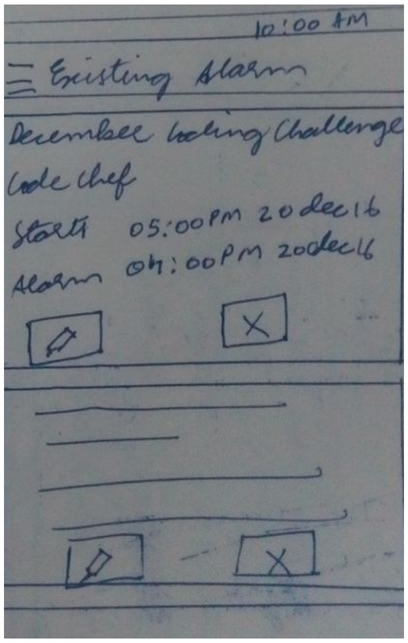
Sample notification for a contest that is about to start. Clicking on the notification, it takes user to detail page of the contest, which has all the details, including the web link to the contest.

Screen 8: Notification Settings



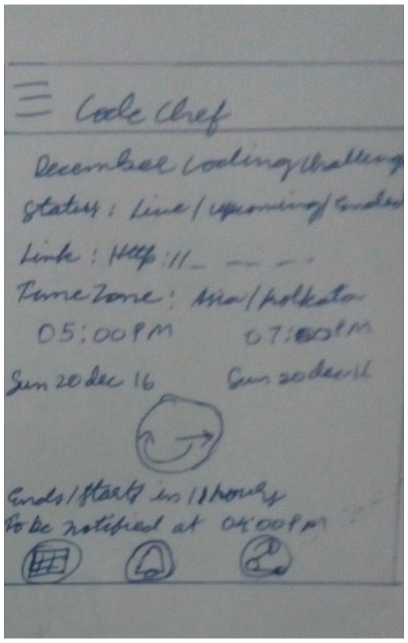
User can select the time for notification about any upcoming event. Also to sound alarm for the same.
User can also select to be notified when we get any event in future that is recently being added to db.

Screen 9: Notification Settings



This screen lists all the reminders that user has set up for various upcoming contests.
User can edit/delete these reminders.

Screen 10: Contest Details



Contest detail page, contains all the information related to a particular contest.

For live contest, it shows a timer which refreshes every second and shows time left in that contest.

Page also contains link to the contest. Also displays user's time zone.

Key Considerations

How will your app handle data persistence?

Describe how your app with handle data. (For example, will you build a Content Provider or connect to an existing one?)

App has local sqlite db for data persistence. DB operations will be wrapped inside Content Provider. Latest data will be fetch from web server by sync adapter periodically, and synced with local db copy using content provider.

Describe any corner cases in the UX.

- Handling fragment history, so that when user presses back button, user is taken back to appropriate fragment. This can be achieved by maintaining fragment stack history with title, and we can pop fragment back when user presses back button.
- UX for phone and tablets will be different, on tablet (large screen devices), list and detail page will be shown simultaneously, so need to handle separate layouts.
- Handling orientation changes.

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

For example, Picasso or Glide to handle the loading and caching of images.

- Retrofit/Gson: For making REST API calls, and converting json response to java objects.
- Picasso for loading Images
- Data Binding: Makes code short a beautiful and reduces boilerplate code.
- Snappy/Realm DB: Provides better performance than SQLite

Describe how you will implement Google Play Services.

- Firebase Analytics and Crash Reporting: For tracking installs, clicks, and what screens are frequently visited by user and also reporting crashes and unhandled exceptions. Firebase analytics provides comprehensive analytics data on its dashboard.
- Google Admob: For advertising and monetization.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Project contains following main components. Create layout for following components according to above mentioned mockups.

Activities:

1. Splash Screen Activity
2. Main Activity

Fragments:
1. Live Contests
2. Upcoming Contests
3. Past Contests
4. Select websites
5. Notification Settings
6. Alarm Lists
7. Detail Fragment

NavigationView for left navigation drawer

Task 2: Implement UI for Each Activity and Fragment

Main Activity will host all the live, upcoming, past and other settings fragment. These fragments will be handled via fragment transactions.
List the subtasks. For example:

- Build UI for Splash Screen
- Build UI for MainActivity

Task 3: Implement DB, Content Provider, Sync Adapter

For data persistence, implement local sqlite db, wrapped in content provider to provide required abstraction, Content Provider will provide methods for CRUD operations. Sync Adapter will handle task of syncing data from web server to local db.

3 Tables are required in DB
1. Contest Table
2. Website Table
3. Notification Table

Need to have join between contest and website table to show only those contest which user has selected in settings.

Also delete data that is more than 7 days old, so as to keep local db light in weight, also that data becomes useless.

Use shared preference to keep track of our sync time, and make next sync after appropriate interval.

Task 4: Implement Notifications, Time Change & Time Zone Change & Boot complete broadcast receiver

To enable user to set up notifications, need to implement Notifications via Notification Manager. Also, need to register for time change, and time zone change broadcast, so as to make time difference correction for existing reminders. So need to listen to time change, time zone change, and also reboot broadcasts.

Task 5: Implement Google Play Services

Implement Firebase for Analytics and Crash Reporting and implement Google Admob for Ads

Add as many tasks as you need to complete your app.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"