**Documentation**

**Application** : Percept of Personality

**Min** **version**: Kitkat (Android 4.4)

**Backend** : Java

**FrontEnd**: XML

**Fragments** :

* **Account** : To show the details of the user (name and email)

1. BackEnd : accountFragment.java
2. FrontEnd : fragment\_account.xml

* **Contact** : To show how the user can reach to moodcafe.

Hyperlinks to the moodcafe website, google map and gmail(with default reciever : contact@gmail.com)

1. BackEnd : contactFragment.java
2. FrontEnd : fragment\_contact.xml

* **Settings** : To show settings for the stories the user has played with a popup menu :
* To update the protagonist name
* To delete current game progress
* To view the final result if the story is completed

1. BackEnd : settingFragment.java

**Functions** :

**storeValues**() : To store the data of all the stories which the user has played in an array list of GamesPlayed class to show at the settings screen. These data includes story title, story Id, current game progress.

**takename**() : To show a dialog box which takes the name and update as its protagonist name in the story.

**confirm**() : A confirm dialog box to confirm the user really wants to delete the current game progress.

**Classes**(**inner**):

**GamesPlayed** : Class to store values for each story played **GamesPlayedAdapter** : To Adapt array of GamesPlayed class to the settings screen.

1. FrontEnd : fragment\_setting.xml

**Classes** :

* **AnalysisDataManger** : Full analysis of the result at the story completion which includes accessing final content in the result screen, applying the logic for different types of result, updating the choices chosen by the user in a json file which is stored in the phone’s memory.

**Functions** :

**AnalysisDataManager**() : constructor

**updateChoice**() : Parsing json string to analysisParser class using gson library and then using that class to add each choice chosen by the user with the help of questionId, questionType and choice number and then storing updated json in the editor(phone’s memory).

**idtoindex**() : To return the position at which the choice has to be updated in the string array with the help of questionId.

In short to return index number against questionId.

**Index**: 0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a1 | b2 | c1 | d2 | e1 |

**questionId**: “a” “b” “c” “d” “e”

**initialiseAnalysis**() : To generate a json string (if it is the first story) or add json object(if it is not the first story played by the user) to the json string generated initially which will be stored in the editor(phone).

**Format** :

**{**

**“Analysis”: [**

**{**

**"story\_id":**1**,**

**“Qt1” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt2” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt3” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt4” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt5” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt6” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt7” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt8” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt9” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**},**

**{**

**"story\_id":**2,

**“Qt1” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt2” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt3” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt4” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt5” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt6” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt7” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt8” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**“Qt9” :** [“a1”,”b2”,”c1”,”d1”,”e1”],

**}**

**…**

**]**

**}**

* **AnalysisParser , StoryDataParser :** Class to parse above analysis json string.
* **ChooseGameAdapter :** Class to adapt stories for the user to choose at the main screen which includes parsing story cover images. After clicking any of the story showing a dialog box regarding resuming or playing the game, the action for which will be controlled by **onClick**() function.
* **Content, Story, MainParser :** Class to parse story.json file.
* **GameResumeParser, GameProgress :** Classes to parse json string ()
* **ResumeManger :** To handle the resume game option in the screen by storing current scene id against each story id.

**Format**:

**{ “values”:[**

**{**

**“sceneId” :** last sceneId at which user stopped playing

**“storyId” :** 1,

**“progress” :** current progress of story in percentage,

**“updatedBackground” :** last background used in the game

before user stopped playing it,

**“protagonistName” :** name with which the user is playing a

story,

**“storyTitle” :** Title of the story

**}**

**…similarly for other stories**

**]**

**}**

* **Typewriter :** This class is an extension of a textView which sets the text in it with a typewriter effect.

**Activities** **:**

* **Login :** As the name says this activity is used for login.

FrontEnd : activity\_login.xml

BackEnd : LoginActivity.java

**Functions :**

**setupGoogleLogin() :** As the name says it is used to setup the google login feature in the app.

**setupFbLogin() :** To setup fb login feature in the app.

* **Signup :** This activity is used for signup

FrontEnd : activity\_signup.xml

BackEnd : SignupActivity.java

* **MainActivty :** This is the main screen of the app, it handles the Hamburger menu(Extending NavigationDrawer Class) ,shows the list of the stories which user can play and also the resume game feature.

FrontEnd : activity\_game\_details.xml (the name means it shows the story details)

BackEnd : MainActivity.java

**Functions :**

**askResume() :** To show the resume dialog screen against each story along with the story description for the user to resume if not playing for the first time and to take protagonist name if playing for the first time.

**convert() :** To parse the game json data : story.json to respective classes and getting the images and story title to show at the screen

**loadJSONFromAsset() :** To convert json file to a string to parse json string to the class easily

**jsonInitialise() :** To initialise the resume json string with each story to have zero progress , zero scene id , living room background, protagonist name, story ID and respective story title.

* **NavigationDrawer :** To handle the functioning of hamburger menu.

FrontEnd : activity\_navigation\_drawer.xml

BackEnd : NavigationDrawer.java

* **Result :** This activity shows the final result at the screen.

FrontEnd : activity\_result.xml

BackEnd : ResultActivity.java

**Functions :**

**updateToServer() :** To update the feedback to the server

**userSatisfaction() :** To update the json with feedback given by the user against each description.

* **Game :** This activity handles the game part.

FrontEnd : activity\_game.xml

BackEnd : GameActivity.java

**Functions :**

**addContentToScreen() :** To add content to the screen at different positions against each scene id. (refer code)

**displayQuestion() :** showing the validity question

**update() :** To save current sceneId so as to continue next time where the user left.

**updateToServer() :** updating the user analysis to the server in the data store API

**someChanges() :** Changing the dialogue of the character little bit, breaking it if it is a long dialogue i.e. if it is more than 3 lines.

**convert() :** parsing story.json in Story and Content class

* **SplashScreen :** To show the slash screen.

FrontEnd : activity\_splash\_screen.java

BackEnd : SpashScreen.java

**Layouts :**

* **gamesplayedlist.xml :** To show played games list in the settings screen of the app.
* **nav\_header.xml :** Header layout for the hamburger menu
* **resume\_dialog.xml :** FrontEnd for the dialog box to show the resume game option.
* **story\_card.xml :** Story cards at the main screen of the app.
* **toolbar.xml :** Toolbar for the hamburger menu at the main activity(screen) of the app

**Menu :**

* **gamessettingspopup.xml :** menu for the settings of each game played which includes change protagonist name, deleting current progress and view result.
* **navigation\_menu.xml :** menu for the hamburger menu

**Raw(json files) :**

* **analysis.json :** It stores the final result analysis that will be showed in the ResultActivity.
* **Story.json :** It stores the content of each story.

**Drawables :**

All the images and some xml files:

* **feedback\_onfocus.xml :** green border background design for the feedback.
* **loginbutton\_selector.xml :** button design for login, signup and various positions in the game.
* **progressdrawable.xml :** design for the progress bars used at various positions in the game.