Taylor Del Matto
Final Project Documentation
CS 496

Android mobile application built in Android Studio using Java with Google Maps API and User Accounts Backend hosted on AWS EC2 server built on Node and DynamoDB (Amazon's NOSQL database)

For this project I implemented a cloud backend on an aws server using node express and dynamodb that contains user accounts information for the mobile application I created earlier this term. The mobile application is the poker app designed to benefit and aid poker players in learning understanding, and finding casinos to play poker. The database contains a Casino Entity containing information about casinos, and a Users entity containing information about User accounts including password, username and other information about that user, mycasinos list. Each User account entry contains a list of that user's favorite casinos. There is a one to many relationship between Users and Casinos; a user can have many casinos. The entire backend was implemented entirely from scratch using dynamodb, node, and express. All of the following data is stored, retrieved updated and deleted by sending http calls with parameters to the specified URIs which then update/retrieve/create/obtain the desired info from the nonrelational database.

The /createusers and /createcasinos take a get request to create the tables in dynamodb used to store the casino and the user account information. Similarly /deleteusers and /deletecasinos are get handlers that delete the users and casinos tables respectively.

The /cloudcompute URI is a post handler that takes information about a hand, and calculates the mathematical odds of a player winning that hand and then responds with a string to be displayed on the client.

The /newuserprofile URI is a post handler that takes a JSON object with attributes password, username, dob, and playerstatus to initialize a new profile entry in the users table in dynamodb

The /login URI is a post handler that takes a JSON object containing the username and password of a user attempting to login. The handler queries the users table for a matching username/password, and if correct, it sends back the information associated with that user account to the client in a JSON string, or an error message if there is no such associated account.

The /updateprofile URI is a put handler that takes a series of account update information supplied by the user on the client device. The handler attempts to update and sends an update status back to the client.

Taylor Del Matto
Final Project Documentation
CS 496

The /deleteprofile URI takes a URL encoded username and deletes the associated account in the users table. It sends a status update string back to the client.

The /getcasinoslist URI is a get handler that takes a username and returns a list of favorite casinos associated with a supplied username.

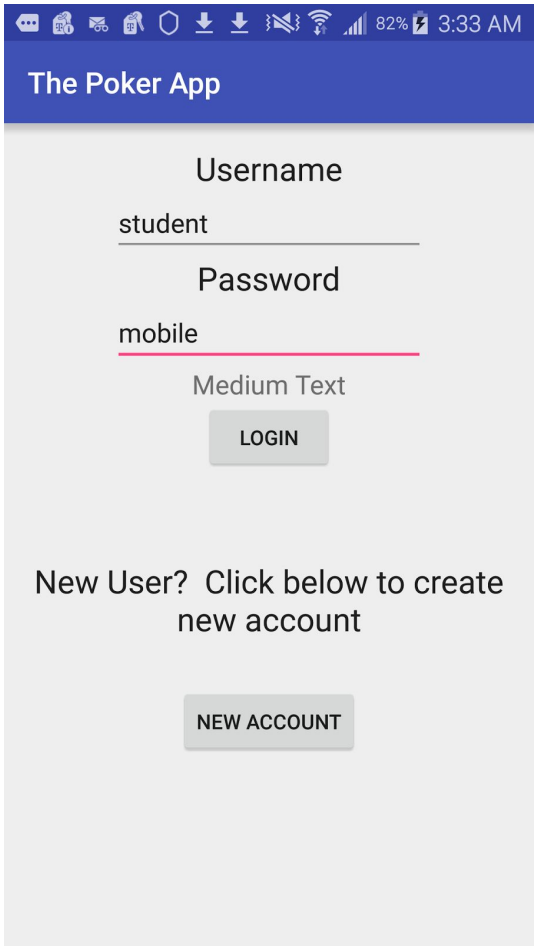
The /addcasino URI is a get handler that takes a username, password and casino name. If the casino exists in the casinos table, It adds the casino name to the mycasinos array in the referenced user account in the users table.

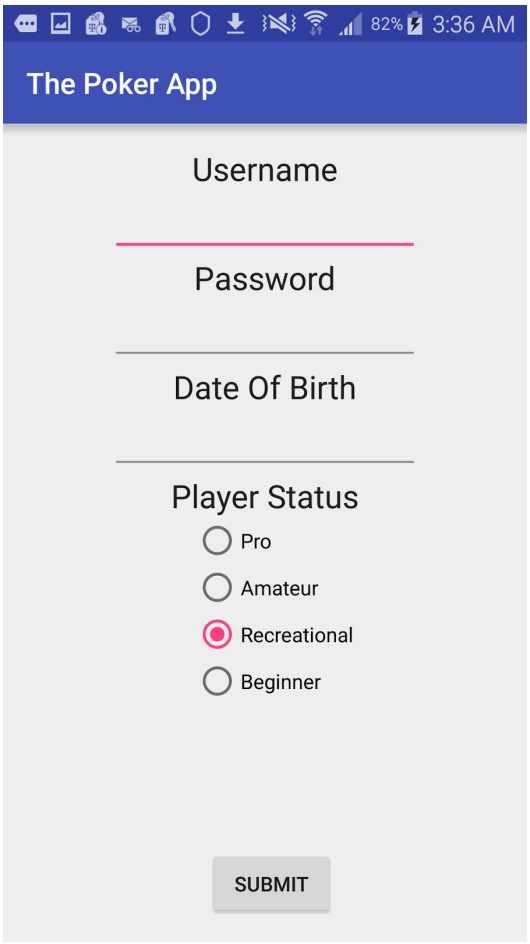
The /removecasino URI is a get handler that that takes a username, password and casino name. If the casino exists in the mycasinos array in the referenced user account, It removes the casino to the referenced user account in the users table. And sends a remove status to the client.

Account System

The opening screen to the mobile app is a login screen with a link to create a new profile for new users.

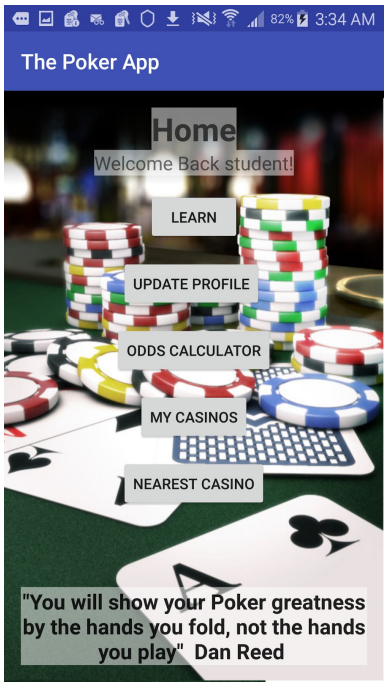
Taylor Del Matto
Final Project Documentation
CS 496



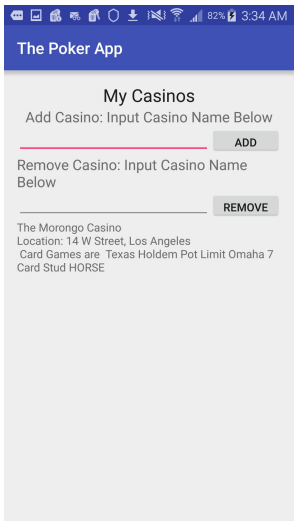


Upon following the link and creating a new user profile, the android app sends the data to the /newuserprofile URI creating an account in the form of an entry in the Users table in the non-relational database. On the login page, the user can now access the app by entering the password and username used originally to create the account. The login button sends a post request to the /login URI. If the login URI queries a match with the sent password/username in the Users table, the URI sends back the associated account data to the client instead of an error message, opening up the main screen of the device and caching the User profile information to pass to the activities.

Taylor Del Matto
Final Project Documentation
CS 496

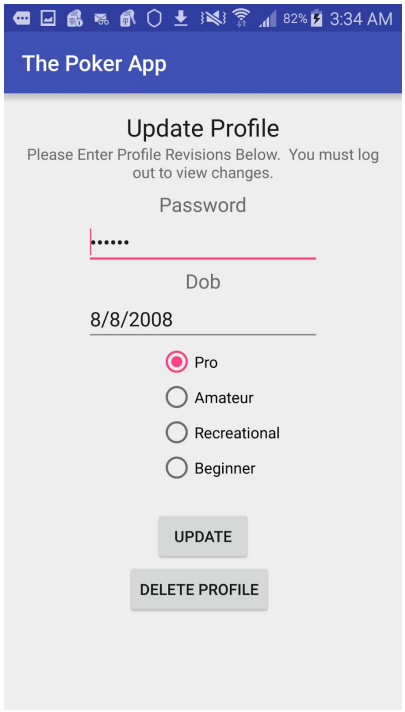


The mycasinos activity then allows a user to add casinos from the casinos table in the non-relational db to his profile, and viewed in the my casinos activity (see video).



The user account information can be updated by accessing the update profile activity which renders the associated account information into a form for an easy one click update. This data is sent to the /updateprofile URI.

Taylor Del Matto
Final Project Documentation
CS 496

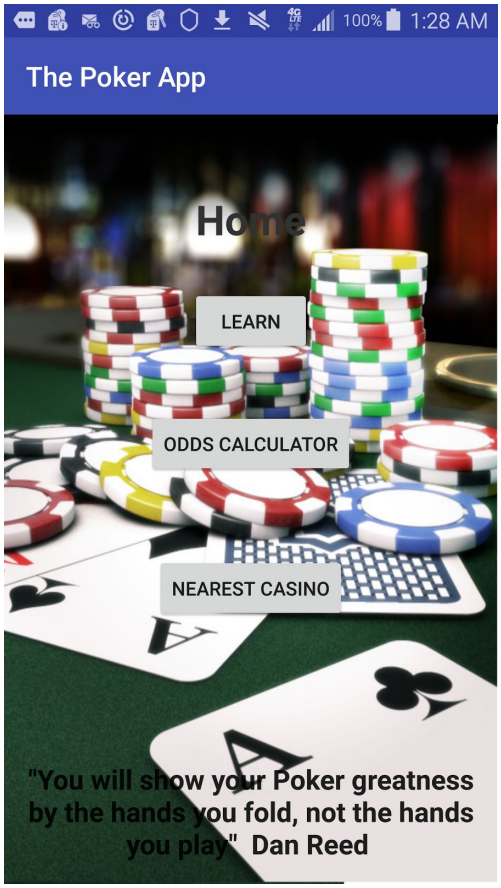


The mobile application uses android volley to make all of the http requests. After the data is retrieved from the login screen it is passed to all following activities. Mycasinos runs its own get query to obtain the specific casino information for each casino in the mycasino array for that users account. Since each User entry only contains a list of the users favorite casinos, a get request to /getcasinos list is necessary to get information for each of those casinos and render it into the mobile application. Org.json library was used to parse the JSON data returned by the server. The data is locally cached and is not stored in files, upon closing the application all data must be retrieved from the server again...and it is upon login.

Here is some information that was already covered in my mobile app assignment 4.

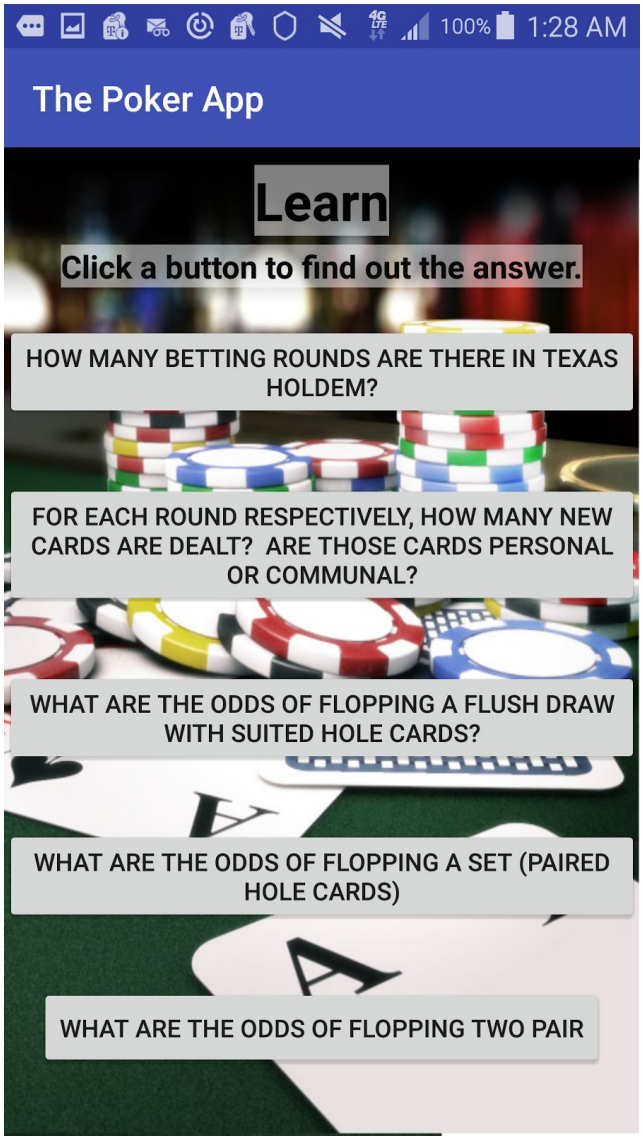
For this assignment, I developed an android app in Android Studio called *The Poker App.* The Poker App has 4 activities. The first is the home screen activity, which essentially just serves as a navigation portal to the other 3 activities. The home screen has 3 buttons which have listeners that start up each activity respectively on click of that button. To navigate back to the home screen, simply click the back button. Navigation between activities is completed by pressing the back button to get back to the home screen and then starting up the desired activity.

Taylor Del Matto
Final Project Documentation
CS 496



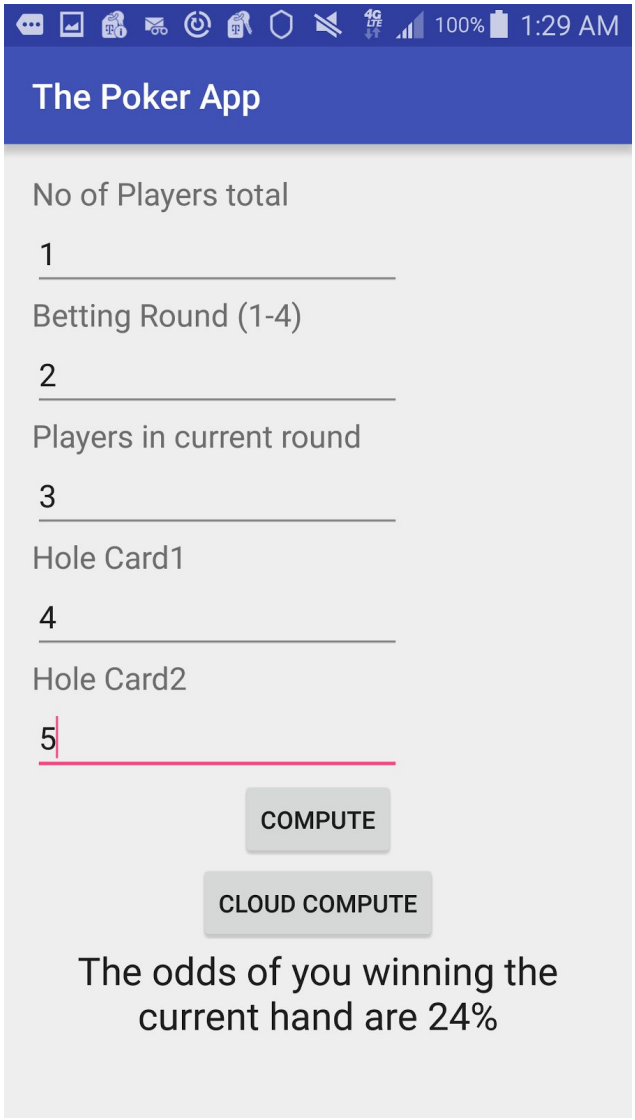
The second activity in the application is the learn activity. The learn activity has a teach yourself quiz on poker knowledge and math facts. A series of questions are displayed on the learn activity. Upon clicking a question, an answer is rendered into a textview below each question. Clicking each question again removes the answer, so that you can quiz yourself multiple times.

Taylor Del Matto
Final Project Documentation
CS 496



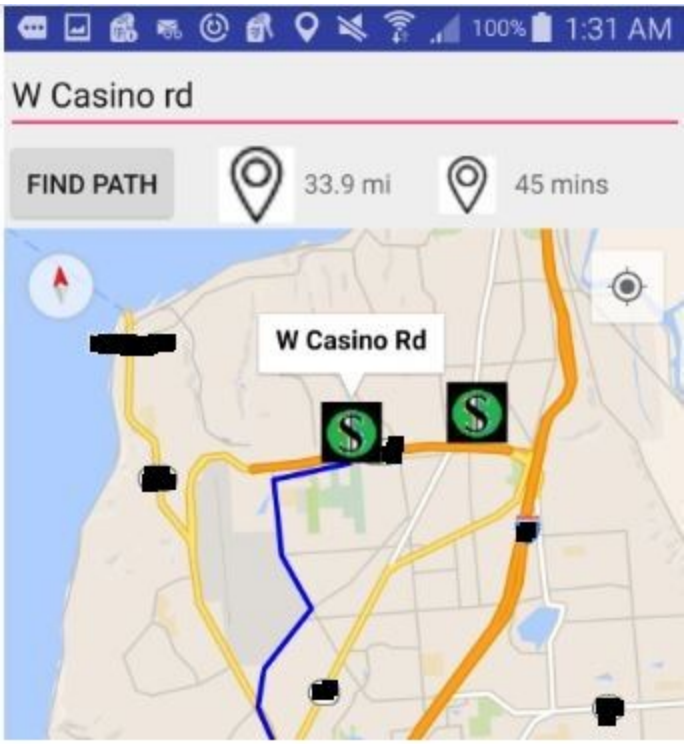
The third activity in the application is the Odds calculator. The odds calculator takes a series of form input parameters based on a given poker situation. Upon pressing one of the two compute options, the odds of winning in that situation are calculated and rendered into a textview below. The algorithm is rudimentary; developing a legitimate algorithm for this purpose could take days, and this is a mobile/cloud development course; not combinatorics. However a real algorithm could easily be substituted here. Two compute options are offered here, a “compute” option and a “Cloud Compute” option. Upon clicking the Compute option, the odds are calculated on the phone’s processor. Upon clicking the Cloud Compute option the parameters are sent in a JSON post request to my Poker Odds Calculating API, which takes the parameters, calculates the odds of winning the hand, and then sends the answer back to the phone, where the response is parsed and displayed.

Taylor Del Matto
Final Project Documentation
CS 496



The fourth activity in The Poker app is the Nearest Casino activity. This activity uses the google maps API to find your location, search for nearby casinos, display those casinos on the map by plotting pins at those locations with name and address descriptions, and then allowing the user to enter that address into the above navigation text entry, and generate driving directions to that location. First, the app uses the phone gps to obtain the user's GPS coordinates. Then taking those coordinates, I used the Google Maps API to center on and zoom in on those coordinates. Then I used the Google maps API geocoder to search for casinos in a 69 mile radius of the current GPS Location obtained earlier. Pins were placed on the casino locations returned by the geocoder with the Addresses and names input as pin onclick descriptions. From here, a pin could be clicked displaying its address. Once the user enters the address into the edit text field above, and submitting, the user can obtain driving directions in the form of a path to the casino. This is obtained by submitting an address to the google directions api which returns a series of points, that are connected by google maps polylines to create a path.

Taylor Del Matto
Final Project Documentation
CS 496



Shown below is the node express handlebars server handler for the cloud compute post request (Activity 3)

```
app.post('/cloudcompute', function(req, res, next){
	var postbody = {};
	postbody = JSON.parse(req.body);
	console.log(postbody);

	var result = (parseInt(postbody.np) * parseInt(postbody.round) *
parseInt(postbody.mc) * parseInt(postbody.pcr)) % 100;
	var results = JSON.stringify(result.toString());
	res.send(results);
});
```

Sources:

I coded everything from scratch in this project except for the modules I used in order to parse the directions from the google directions API. After building around those modules extensively using the google maps api for android, location, and the google maps geocoder, i did not find it important to write my own implementation, since there was a tutorial to build the modules online that I followed, (but did not internalize it). Anyways, I could have simply created a more

rudimentary version of my directions finder which finds directions between current location and a nearby casino by simply creating a linear path between the locations instead (which would have been suitable for this project). But I did not see any point in doing that. So just to make it clear, I wrote the modules directionfinder, directionfinderlistener, distance, duration, and route by closely following this tutorial. https://www.youtube.com/watch?v=CCZPUeY94MU and they are not unique enough to claim as my own code. These libraries only aid the path creation in the findcasino activity. Outside of that I implemented all 10 activities and their xml files entirely independently, including all http calls, and I implemented all cloud data storage and non relational database interaction entirely from scratch.