We have used Model View Controller (MVC) pattern in our system. We chose to implement MVC pattern because it is more suitable for what we are trying to implement. The MVC pattern splits code into one of three MVC components.

* **Model**: which are the classes that represent the data to be stored. It interacts with the controller and has no knowledge of the interface. For example, Reservation, User, Spot, etc.
* **View**: contains what is visible to the user (or allows the user to interact with elements on the screen). For example, all the xml files in the application.
* **Controller**: interacts with the view and the model by responding to user input and retrieving data the user requests. It also manages the application logic. For example, LoginActivity, SignupActivity, ReservationActivity, ShowReservationActivity, etc.

A simple way to think about our application structure is the following:

The user interacts with the view (user interface) then the Controller handles the user input and transfers the information to the model. The controller does all the computation and processing between the model and taking the inputs and showing the output from and to the view.

View

Controller

Model

Advantage of using this pattern:

* Separation of Concerns:

MVC allowed us to separate our logic from our presentation layer. This “separation of concerns” allowed us to quickly find and edit portions of our code whenever needed for example if a miss or a fail happened in one of the algorithms, we can easily re-write it without having to interrupt the user interface.

* Re-usability and Flexibility:

The UI can be completely changed without touching the model in any way. Additionally, using MVC might help in increasing the scalability of the application.

* High Cohesion and Low Coupling:

Applying such principle making our system high cohesion in the way that all functions in a single class are functionally related and preform methods that complete a specific job. Also, it makes our system low coupling which mean changing something major in one class should not affect the others.