Birla Institute of Technology & Science, Pilani
Work Integrated Learning Programmers Division
First Semester 2022-2023
Assignment 2
Weightage 10%

Submitted By**: Satish Kumar Sharma**
BITS ID: **2022mt93327**

Assignment 2 (10% Marks)

Assignment #2 (10% weight)

Objective: To gain experience in architecting real life applications in domains such as Retail, Transportation, Healthcare, Hospitality, etc. Example systems: Swiggy, Uber, an IoT system to monitor health of industrial air conditioners.

Activity

1. Identify top 3 Architecturally Significant Requirements (ASRs) and write them in the form of a Utility tree. Why are these architecturally significant?
2. Describe in detail, the tactics you recommend for each ASR. For example, if caching is a tactic you recommend, please mention what you will cache, what tool you would use, how it will work, etc.
3. Draw 2 software architecture diagrams – component & connection view and deployment view – to understand how the system works.
4. Indicate important messages between components by labelling the connections in the C&C view. Also indicate the communication method used.
5. Draw sequence diagram for one major scenario (use case). Mention the scenario.
6. State the architecture patterns used. Explain, where in the architecture, these patterns have been used.
7. What did you learn by doing this assignment? Mention 3 key learnings. One slide per person.

Evaluation criteria:

a) Easy-to-understand diagrams
b) Clarity of description
c) Correctness of work products
d) Participation in the Discussion Forum (Maturity, Discussion, Response)

Answer:

GOALS OF MOBILE BANKING SYSTEM:

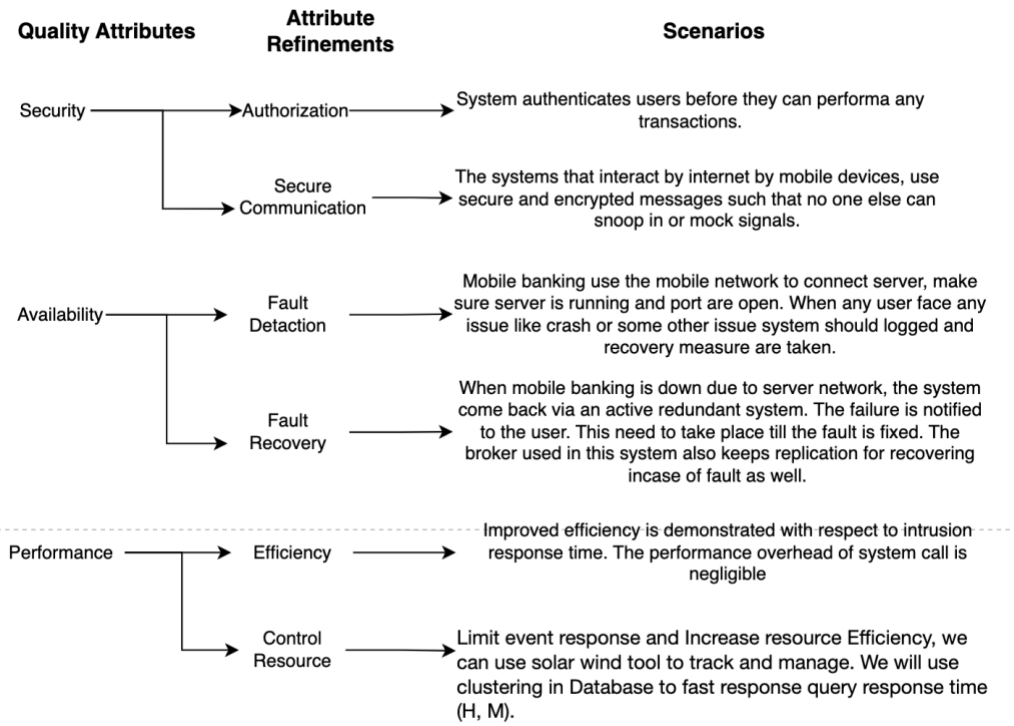
- A Mobile banking is a financial institution which is involved in borrowing, Fun transfer, Utility payment and lending money.
- Mobile banking **allows consumers to be able to access banking services from anywhere**.
- Banks take customer deposits in return for paying customers an annual interest payment. The bank then uses the majority of these deposits to lend to other customers for a variety of loans. The difference between the two interest rates is effectively the profit margin for banks. Banks play an important role in the economy for offering a service for people wishing to save.
- Banks also play an important role in offering finance to businesses who wish to invest and expand. These loans and business investment are important for enabling economic growth.

I have considered the mobile banking system, which is very helpful now a days to easily access your bank account and perform transactions and utility payments.

Top 3 Architecturally Significant Requirements (ASRs)

1. Security
2. Availability
3. Performance

Utility tree of Top 3 Architecturally Significant Requirements (ASRs):

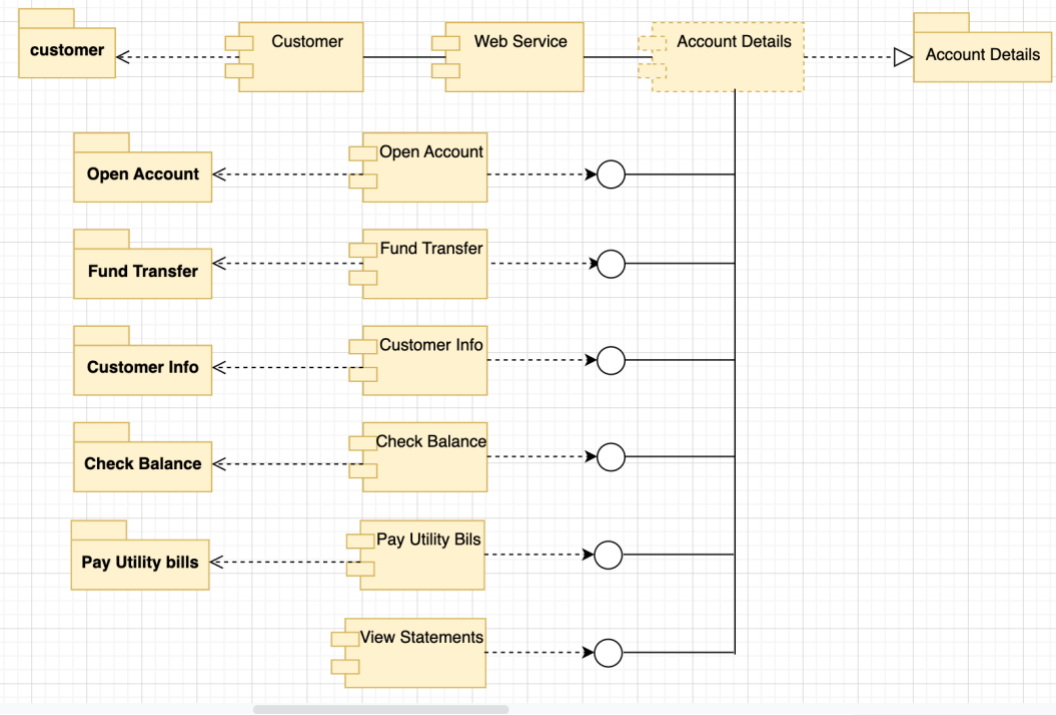


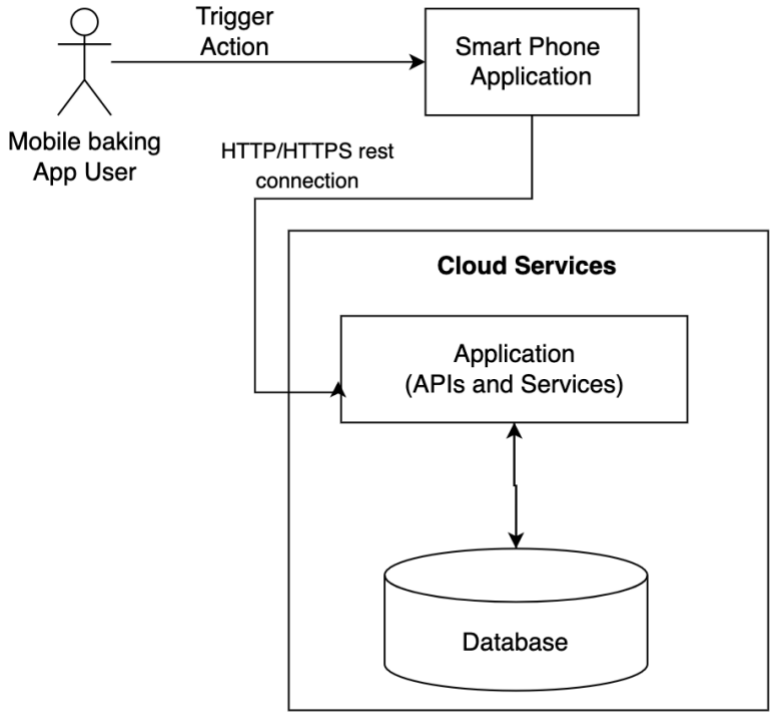
TACTICS TO ACHIVE TOP 3 ASRS (CONT.)

Quality Attribute	Scenario (Attribute Refinement)	Tactics
Security	Only authorized users can interact with the mobile banking system by app. Role based permissions need to be supported a well.	• The hosted cloud application will handle requests to the backend server/DB after authenticating with SSO (Single sign on). Other measures such as two factor authentications can also be used. This can be done by integrating to OAuth. • The application can have Users be put onto admin or other roles groups which help to segregating the kind interactions.
	The system support https network should use encryption supported to make sure communication channels are secure	• Any communication between mobile app and systems must be encrypted with the network key or other tokens created while authenticating the user. • Keep the device information against every user name to tract the login to prevent unauthorize access.
Availability	Detect faults with any systems or apps in the Mobile baking system.	• The system will check for the availability of the systems by employing ping messages. When there is a failure in response after a fixed number of retries then fault recovery can be triggered (This can be done activating the backup redundant system). The system can also notify the user of failures in the system by notification. • The hosted application should scale up on demand and not fail due to faults related to high traffic.
	Recover faults when it is detected in any system or hub within the system	• When mail server fails, then we can employ active redundancy, where a new server can come up soon and resume operations and notify the user of the failure. • The broker used for communication should also be replicated, So that on a failure scenario. • The application hosted would be also need to employ redundancy to make sure cloud services are highly available and use deployment strategies which aid in scaling up on demand.
Performance	The system should available 24/7.	• Down time should be less than 99.999%. • Limit event response and Increase resource Efficiency, we can use solar wind tool to track and manage. We will use clustering in Database to fast response query response time (H, M). • Monitor system and Self testing, Security is low when an attacker can decrease availability of the system (denial of service attack (DoS)).

COMPONENT & CONNECTION VIEW DIAGRAMS:

Below diagram display the internal component inside the API services.





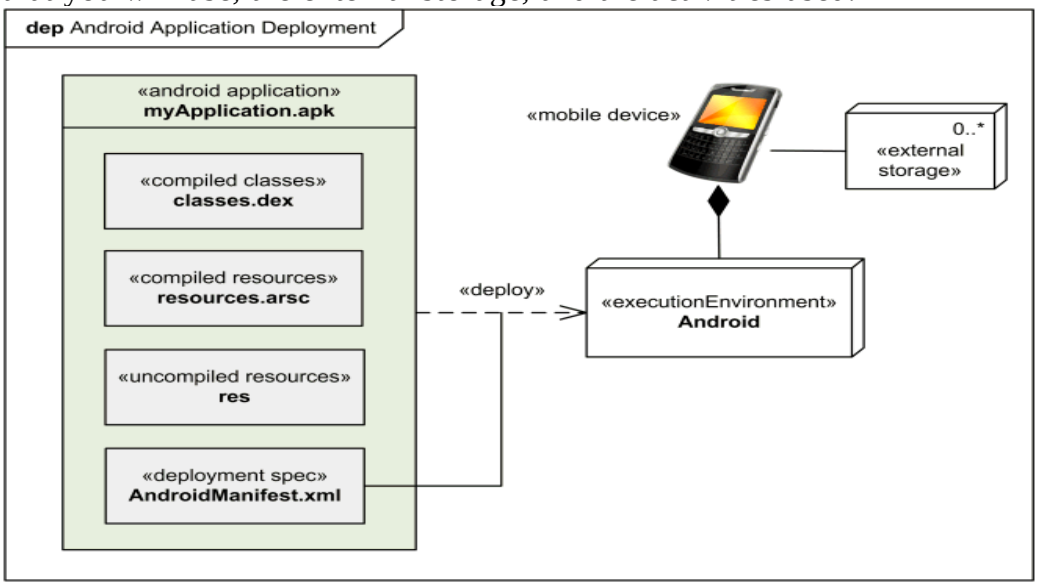
The following are the interactions that are seen between the major components in the system:

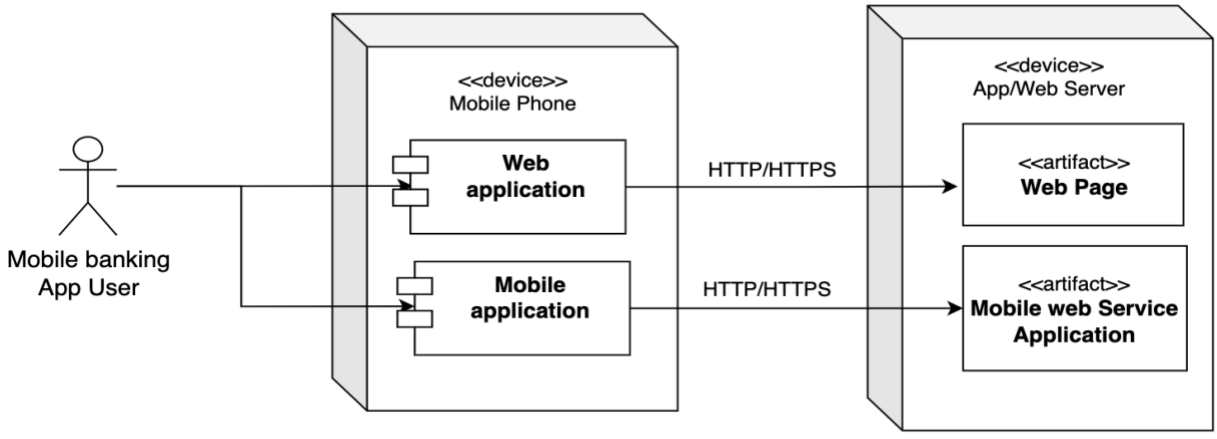
- The Smart phone application can send instructions via the cloud hosted services from a remote network or using WIFI or mobile data from mobile application.

- Once trigger initiated from application, API services will get call by HTTP/HTTPS protocol from the mobile app.
- API services will save data or perform business logic and return the response to Mobile application then app will display message to user.

DEPLOYMENT VIEW DIAGRAMS:

This is a simple UML deployment diagram. It tells about the functioning of the mobile banking app and the resources that it would need. The mobile image shows the device that you will use, the external storage, and the activities used.



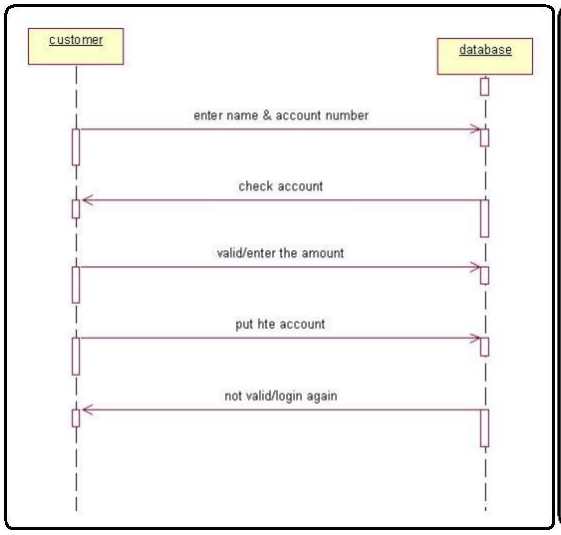


SEQUENCE DIAGRAMS:

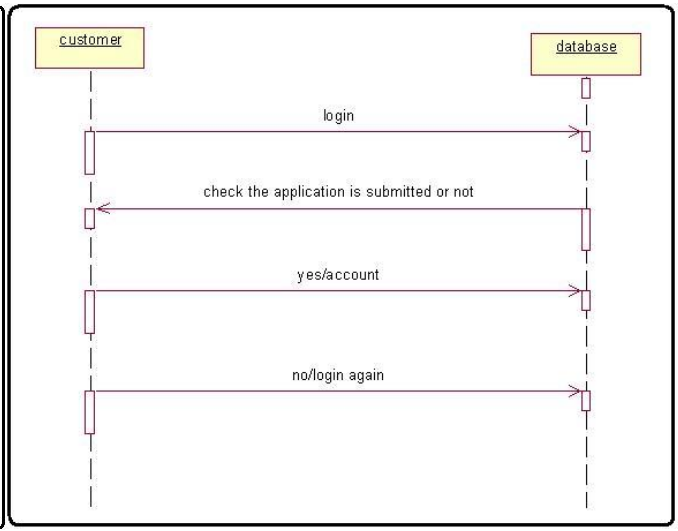
These below sequence diagram shows the use case of how rules are set for the Mobile Banking System. I am showing few major features sequence diagrams.

A sequence diagram has 2 dimensions,
(i) Vertical dimension – represent time
(ii) Horizontal dimension – represents different objects

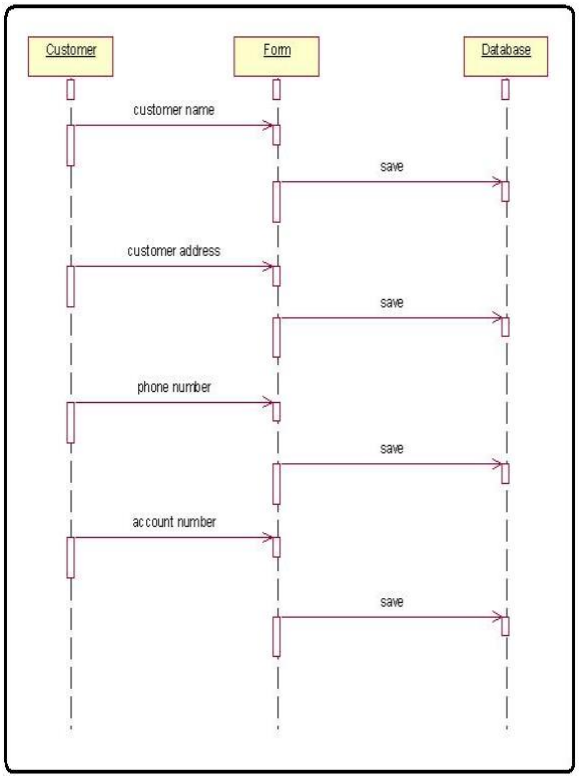
Sequence diagram of Login



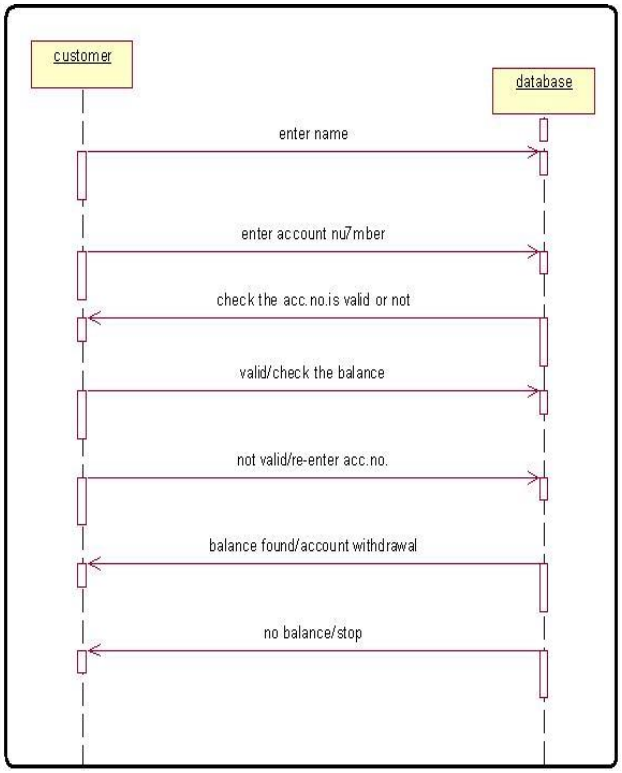
Sequence diagram of Open Account



Sequence diagram of Customer info



Sequence diagram Check Balance



ARCHITECTURE PATTERNS USED

The following are the architecture patterns used in designing the Mobile banking system:

Client Server Pattern: This pattern is used in the cloud hosted application which handles the requests by mobile app users. The server would act as a bridge between the mobile app and the respective server application. The application should be capable of scaling up when the requests it receives increases. All the Mobile banking API call from API rapper will communicate the server by using this pattern.

Model-View-Controller Pattern: The model-view-controller (MVC) pattern separates application functionality into three kinds of components. I have used MVC pattern to separate user interface, Data and the business logic for applications features like fund transfer. Each feature will have their own view controller and View and Model. Model will be responsible to call web services and save response. View is responsible to render or display UI component and take user action. View controller will be responsible to manage view and model based on event trigger from view.

Layering Pattern: The layered pattern divides the software into units called layers. Each layer is a grouping of modules that offers a cohesive set of services. I have used layer pattern to separate application functionality like fund transfer, utility payment, Balance display, Statements... etc.

Broker pattern: The broker pattern separates users of services (clients) from providers of services (servers) by inserting an intermediary, called a broker. I have used API rapper as Broker pattern to consume web services from mobile application. API rapper will call the server to perform the transactions, save detail and get account balance…etc.

KEY LEARNINGS

Personal Learnings

- Gather and find the goal of System.
- Find Architecturally Significant Requirements (ASR) and Prepare Utility Tree.
- Understand about architecture and there uses.
- The component connection diagrams help me to visualize implementation strategies much earlier in system designing and makes planning much easier. The deployment view gives a clear image to operation teams on how the application modules are deployed and helps in rough estimation of cost early on and eases planning.
- Sequence diagrams for different use cases give developers a clear idea on what is expected from the system and all the players involved in that use cases. This will also ease development planning.

Learning to organization

Challenges in building and managing high scalable mobile banking applications system.

Thank you