Design Document

Title: Sick Pro App

Kritika Choudhary
Nimish Bongale
Mahima M A
Mrudoola S

Overview

The objective of this document is to build an application that serves as a personal assistant especially when the user is unwell(or even otherwise).

This application provides a way to consult a doctor of the user's choice based on his/her requirements and schedule an appointment. The application can further help the user in scanning the prescription and sending it to the chemist to purchase the medicines etc. Apart from these, this application also provides a way to hire a maid/nurse/PA (personal assistant) to help him out on a time basis. This application can also be used as a means to hire an ambulance/a taxi as per the requirement. Lastly, the application can also be used to play their favourite games or watch videos etc to refresh themselves.

Case Study

EdPlace

EdPlace is devoted to making parents self-dependent in monitoring their child's educational progress.This app required uploading and downloading content in various sized files. The worksheets were meant to be used by children between 5 and 16 years. We needed to add functionality to prevent writing a lot of code over and over again.(https://www.netsolutions.com/casestudy-edplace)

Sadara

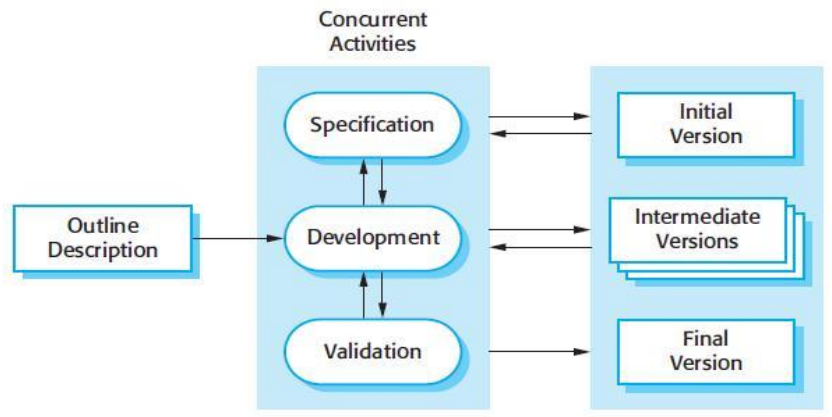
One of IMC's clients, a major bank based in Dubai, was using printed pamphlets to inform customers about available deals and offers. This was not an effective method, as customers did not carry the pamphlets with them all the time. In addition, customers did not remember the deals and offers received through local and digital marketing, thereby missing out on lucrative offers available in the areas they were visiting, passing through or, indeed, living in.
The bank wanted IMC to provide them with a mobile app by which they would inform their customers of the deals or offers available to them wherever they might be.(https://www.netsolutions.com/casestudy-sadara)

Open Source Development

We will be contributing by creating a public repository on github, welcoming issues and reviewing pull requests in order to make our app function better. We also look forward to contributing to other Open Source Projects along the way.
Link to the repository:- https://github.com/nimishbongale/SickPro (currently private)
Others we will contribute to:- https://github.com/keyboardsurfer/Crouton
https://github.com/tokudu/AndroidPushNotificationsDemo
https://ifttt.com/

Software Process Model

Incremental Model:



Using this model the product is designed, implemented and tested incrementally until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements.

Advantages: The cost of accommodating changing customer requirements is reduced.More rapid delivery and deployment of useful software to the customer is possible.
Disadvantages: System structure tends to degrade as new increments are added.

Functional Requirements

The task here is to list out the functional requirements from the admin point of view, to develop an end to end solution for the problem.
Tech Stack involves:- OpenShiftUI, Android Studio, Emulator

1. Login/Signup:- To make the essential details of the user available at all times. Implement a forgot password feature.
2. List of Consulting Doctors:- To have a custom list mentioning all the necessary details of the doctors being consulted.
3. Search feature:- To search among doctors, medicines, chemists and previous records of illness.
4. Update profile:- Based on history of illness, susceptibility and emergency cases.
5. Direct UPI payment to cashier:- Medical services could be paid for directly using any UPI via the App, may it be at the hospital or the local chemist.
6. Call Taxi/Ambulance:- Inbuilt call based booking system for faster access.
7. Reviews/Suggestions:- To build on the app to make it better.
8. Help Chat:- With google assistant features inbuilt with translations.
9. Notes/Notifs Features:- To keep reminders of medicine timings, home remedies, checkups etc.
10. Read & Write into Database.
11. Scan prescription, and send it to the preferred chemist.
12. Play your favourite games & videos to cheer you up.
13. Call maid/nurse/PA on time basis.

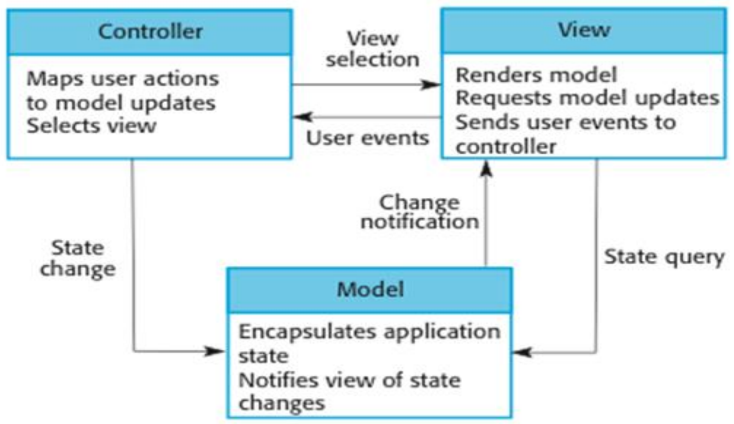
Non-Functional Requirements

These are certain non-functional requirements to be taken care of by the application, to provide a user friendly environment:

1. Ease of use: the users shouldn't face any difficulty in using the app also the users shouldn't find it difficult to understand how the app works.
2. Performance: the response and refresh time should be less.
3. Size: the application shouldn't be bulky but should be of a reasonable size.
4. Reliability: the probability and the rate of failure should be less.
5. Robust: the time taken for recovery from the failure should be less.
6. Portability: the app should be compatible with most of the platforms.
7. Maintainability: the system should be easily be maintained i.e., it should be easy to modify, correct or add new features without making major changes to the system architecture.
8. Security: any transaction made should be secured.

Architecture pattern

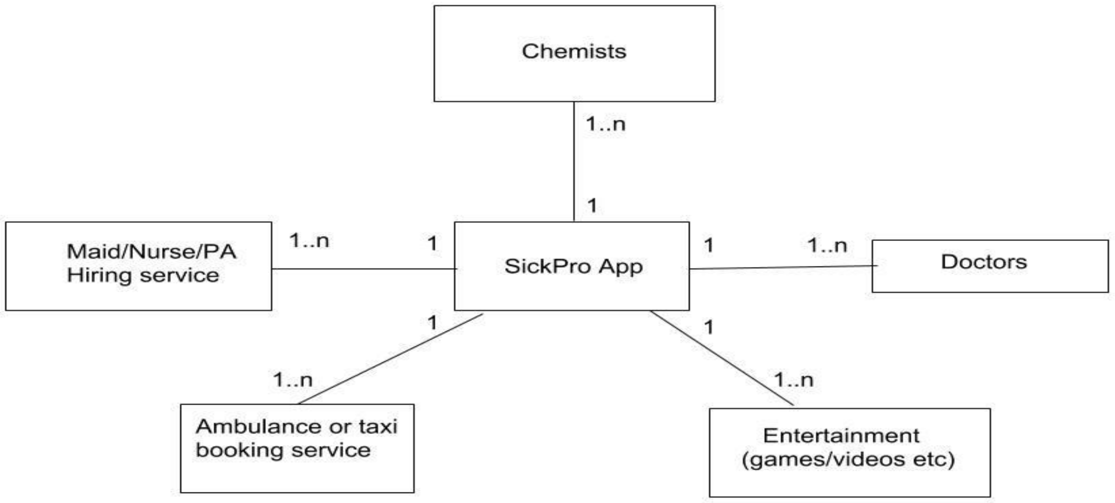
Model View Controller Pattern:



The MVC pattern separates the presentation and the interaction from the system data. The system is divided into 3 logical components that interact with each other. The model component manages the system data and the associated operations on that data. The new component defines and manages how the data is presented to the user. The controller component manages both the view and the model components.

Advantage: This architecture pattern allows the data to change independently without considering its representation.
Disadvantage: Can involve additional code and code complexity when the data model and interactions are simple.

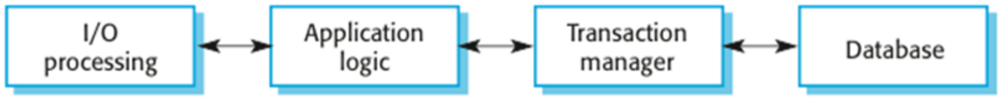
System Context Model


Chemists
1..n
1
Maid/Nurse/PA Hiring service
1..n
1
SickPro App
1
1..n
Doctors
1
1..n
Ambulance or taxi booking service
1
1..n
Entertainment (games/videos etc)

Application Architecture

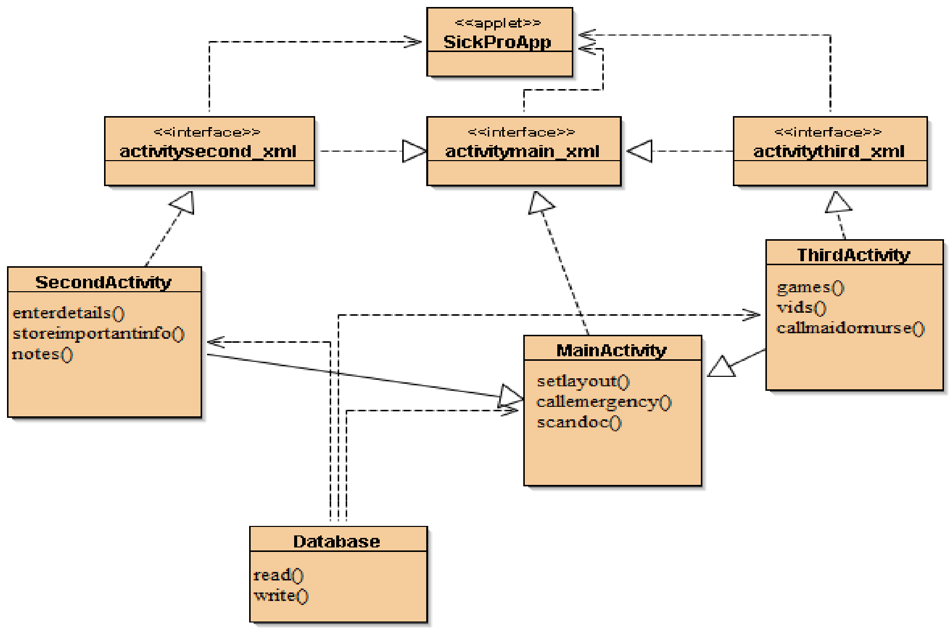
Transaction Processing Application:

These are database-centered applications that process user requests for information and update the information in a database.The user actions can’t interfere with each other and the integrity of the database is maintained.



Transaction processing systems are designed to process user requests for information from a database, or requests to update a database.These systems are usually interactive systems in which users make asynchronous requests for service.

UML Diagram



Design Patterns

Abstract Factory: The abstract factory pattern is a design pattern that allows for the creation of groups of related objects without the requirement of specifying the exact concrete classes that will be used. One of a number of factory classes generates the object sets.

The system will handle a large no. of users(objects) that will perform related but different activities(factory class).