Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |

Note: When you include figures

* Use figure numbers and figure captions.
* Use diagrams/ images/ screen shots with high resolution to get a clear figure
* Use the figure captions in the form of Figure 1. <<caption>> and when explain it in the text, use the abbreviation “Figure 1,” even at the beginning of a sentence.
* When you use a tool to draw diagrams, change the font settings of the diagram; It is better to have font in black colour/ large in size (12pt) and if possible do not fill the objects/ elements in the diagram with a colour (keep the background colour white, for clear visibility)
* Also, try to re-locate the objects closely in a way that take less space (you may drag elements/objects close to each other).
* This will be useful to get a clear image; in order to make the diagram small in size without reducing the resolution quality.
* When you include images in your reports, make them “in line with text” ( picture tool bar 🡪 wrap text 🡪 in line with text) and include the caption accordingly. If you include two or more images together ( in a row), group them.
* Describe each diagram with few sentences.

When you draw diagrams (eg. Sequence diagram) do not include only two object called “user” and “system”. Include all the internal objects within the system, without considering the system as a black box. For example: for a mobile application the main system may consists of sub objects such as , <<UI>>:main\_Interface, :controller, <<UI>>:analysis\_Interface, :local\_DB, etc. (this is only an example; use your own terms).

References:

* Indicate the tools you have used to draw the diagrams

Useful theory for design diagrams : (relationships in Class diagram)

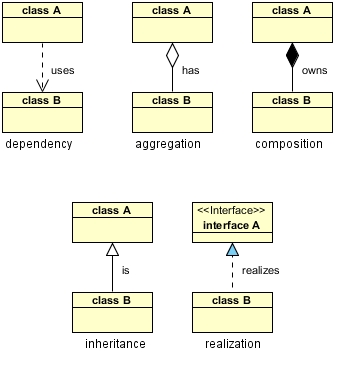


Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 5

4. Use-Case View 5

4.1 Use-Case Realizations 6

5. Logical View 11

5.1 Overview 11

5.2 Architecturally Significant Design Packages 11

6. Process View 11

7. Deployment View 11

8. Implementation View 11

8.1 Overview 11

8.2 Layers 11

9. Data View (optional) 12

10. Size and Performance 12

11. Quality 12

# Introduction

This **Document** provides an overview of the “Smart Planner” application. It includes different perspectives of the application being developed.

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. The specific audiences for the document is identified, with an indication of how they are expected to use the document.

[This section defines the role or purpose of the **Software Architecture Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document is identified, with an indication of how they are expected to use the document.]

## Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

This document will describe the “Smart Planner” application in more detailed and in an architectural manner. The developer can use this a manual to get a clear understanding of the application and to implement the system. Further the clients and the other stakeholder can get a clear understanding of the system being developed by referring this document.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | Person who use the Android application to schedule tasks |
| Server | Web server coded using PHP |
| System | Web server together with the Android application |
| MVC | Model View Controller |

## References

[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

- Indicate the tool you have used to draw the diagrams

## Overview

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

The rest of the document includes Architectural representation of the system by dividing it into logical view, process view, implementation view, deployment view and use-case view. The above views will be described with the use of relevant diagrams. The architectural goals and constraints will also be described. Later the size and performance and the quality of the system will described.

# Architectural Representation

[This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

This section describes the software architecture of the “Smart Planner” application with respect to several views. They are,

* Use-case view

This view describes the main set of functionalities of the system using use-cases and scenarios. The interactions between the users with the system which is considered as a black box will described in this view.

* Logical view

This view describes how the system is structured. the model will be used to understand the system properly.

* Process view

This view describes how the process flow within the system works. With or without the interaction of user an activity will start. Then the data and instructions will be exchanged within the components of the system. This view will consider detail about those processes.

* Deployment view

This view describes the environment which the final system will function. The specific requirements of the environment will be considered here.

* Implementation view

This view describes about the necessary components and how they are bundled to be needed for the implementation of the system.

# Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

Identify sufficient number of use cases for your system. (eg. Less than five use cases are not sufficient for the scope of the project)

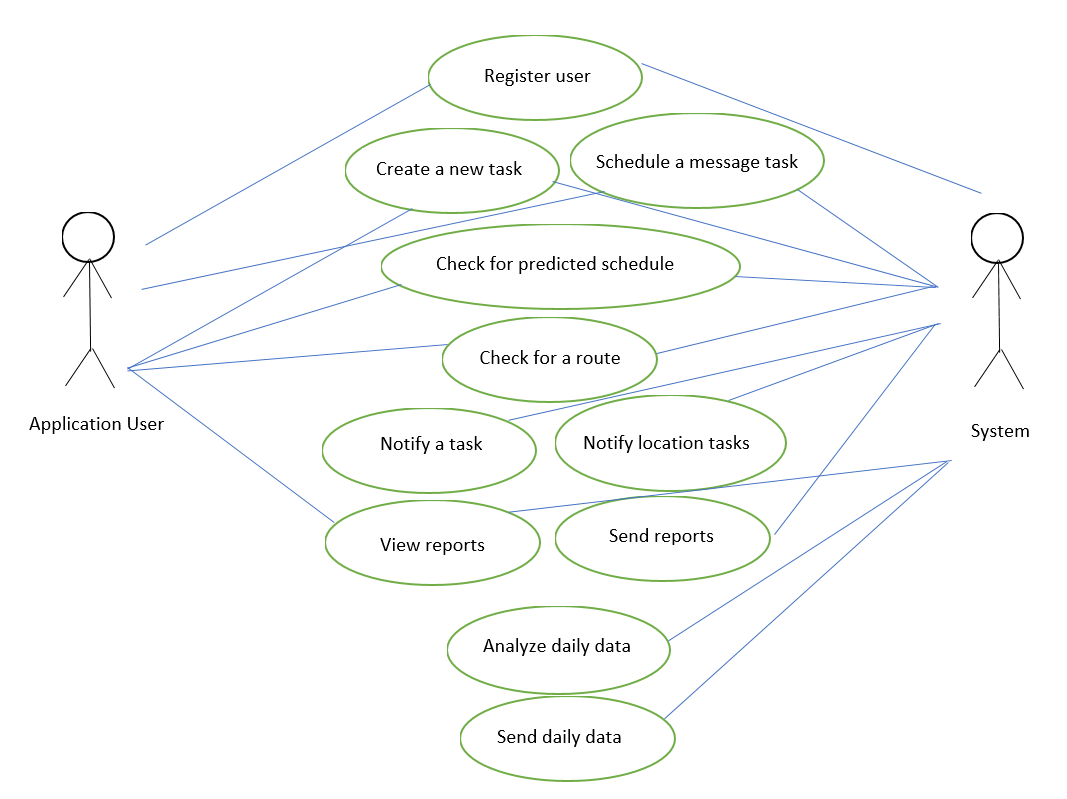
This section describes all the significant use-cases of the system by using the relevant use case diagrams.

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

**Include usecase diagrams and their scenarios.**

For each important/ main/ selected usecase include the realization of a scenario. For example :



### Register the user

|  |  |
| --- | --- |
| **Use case name** | Register the user |
| **Actor** | Application user |
| **Description** | Authenticate user via a google account and creates an account for the user in the system. |
| **Preconditions** | User should have a google account |
| **Main flow** | User selects a google account from the accounts available for him/her. The Android application sends the user email address to the web server. Web server creates an account for the user. |
| **Successful end/post condition** | Registration confirmation message will be visible in the Android application |
| **Fail end/post condition** | Registration failed message will be visible in the Android application. |
| **Extensions** | N/A |

### Create a new task

|  |  |
| --- | --- |
| **Use case name** | Create a new task |
| **Actor** | Application user |
| **Description** | Schedule a new time-based or location-based task in the Android application. |
| **Preconditions** | User should have registered to the system |
| **Main flow** | User enters the task description. Then he/she chooses whether the time-based or location-based option. Then the date, time, location, range of the task is entered. User saves the task. Then the system enters the task to the local database. |
| **Successful end/post condition** | New task added confirmation message will be visible in the Android application. The application will direct to the main view. |
| **Fail end/post condition** | Provided details are false. |
| **Extensions** | N/A |

### Schedule a message task

|  |  |
| --- | --- |
| **Use case name** | Schedule a message task |
| **Actor** | Application user |
| **Description** | Schedule a new message to a contact |
| **Preconditions** | User should have registered to the system |
| **Main flow** | User enters the message body, selects the contact from the list or enters the phone number. Message task is added to the local database. |
| **Successful end/post condition** | Message scheduled confirmation message will be visible in the Android application. The application will direct to the main view. |
| **Fail end/post condition** | Provided details are false. |
| **Extensions** | N/A |

### Check for predicted schedule

|  |  |
| --- | --- |
| **Use case name** | Check for predicted schedule |
| **Actor** | Application user |
| **Description** | Check for the predicted schedule for a given date. |
| **Preconditions** | User should have registered to the system |
| **Main flow** | User provides the date to find the predicted schedule. The Android application sends a request including user email address and the date to the web server. The web server check the remote database for the past data, process the data and return the schedule to the application. |
| **Successful end/post condition** | Predicted schedule will be visible on the Android application |
| **Fail end/post condition** | Server failure or internet connection is lost |
| **Extensions** | N/A |

### Check for a route

|  |  |
| --- | --- |
| **Use case name** | Check for a route |
| **Actor** | Application user |
| **Description** | Find the route for a given location on the map |
| **Preconditions** | User should have registered to the system |
| **Main flow** | User enters the target location. The system finds the route using a google map. |
| **Successful end/post condition** | The route will be visible on the Android application |
| **Fail end/post condition** | GPS is turned off or internet connection is lost |
| **Extensions** | N/A |

### View the reports

|  |  |
| --- | --- |
| **Use case name** | View the reports |
| **Actor** | Application user |
| **Description** | View the task completion details about the user for given period |
| **Preconditions** | User should have registered to the system |
| **Main flow** | User enters the period to find the reports. The Android application sends a request to the web server including the user email address and the period. The web server retrieve the report details from the remote database and sends back to the Android application. |
| **Successful end/post condition** | The reports will be visible on the Android application |
| **Fail end/post condition** | Server failure or internet connection is lost |
| **Extensions** | N/A |

### Notify a task

|  |  |
| --- | --- |
| **Use case name** | Notify a task |
| **Actor** | System |
| **Description** | When a time arrives or the user arrives to the specified location, a notification appears in the phone showing the scheduled task description |
| **Preconditions** | User should have scheduled a task |
| **Main flow** | Once the alarm is fired, the system retrieve the task details from the local database and view it to the user via a notification |
| **Successful end/post condition** | A notification will appear in the phone |
| **Fail end/post condition** | GPS not functioning |
| **Extensions** | N/A |

### Notify location tasks

|  |  |
| --- | --- |
| **Use case name** | Notify location tasks |
| **Actor** | System |
| **Description** | When the user arrives at a specific location, all the task which are scheduled in this place will be notified to the user. |
| **Preconditions** | User should have scheduled a location-based task |
| **Main flow** | Once the location alarm is fired, the system retrieve the tasks details from the local database and view it to the user via a notification |
| **Successful end/post condition** | A notification will appear in the phone |
| **Fail end/post condition** | GPS not functioning |
| **Extensions** | N/A |

### Send daily data

|  |  |
| --- | --- |
| **Use case name** | Send daily data |
| **Actor** | System |
| **Description** | At the end of each day the details regarding tasks which are scheduled by the user are sent to the web server. |
| **Preconditions** | Internet connection must be available |
| **Main flow** | At the end of the day (11.59 PM), All the scheduled tasks of the day with their completed status will be sent to the server. |
| **Successful end/post condition** | Server will send a confirmation request to the application |
| **Fail end/post condition** | Internet connection is not available or server failure |
| **Extensions** | N/A |

### Analyze daily data

|  |  |
| --- | --- |
| **Use case name** | Analyze daily data |
| **Actor** | System |
| **Description** | When a set of daily data arrive to the web server, it analyzes data and store it within the database. |
| **Preconditions** | A set of daily data must arrive to the web server |
| **Main flow** | First the daily data is stored in a separate table in the remote database. Then, the server compare those data with the past data stored in the remote database. Then the predicted schedule data are updated within the remote database. |
| **Successful end/post condition** | A report will be sent to the user via an email |
| **Fail end/post condition** | Server failure |
| **Extensions** | N/A |

### Send reports

|  |  |
| --- | --- |
| **Use case name** | Send reports |
| **Actor** | System |
| **Description** | At the end of each day, week or month, a report about the past scheduled tasks will be emailed to the user. |
| **Preconditions** | The daily gathered data must be analyzed |
| **Main flow** | When the data analyzing is done within the server, it will generate a report including the details of task completion. Then this report is emailed to the user. |
| **Successful end/post condition** | An email will be sent to the user |
| **Fail end/post condition** | Server |
| **Extensions** | N/A |

# Logical View

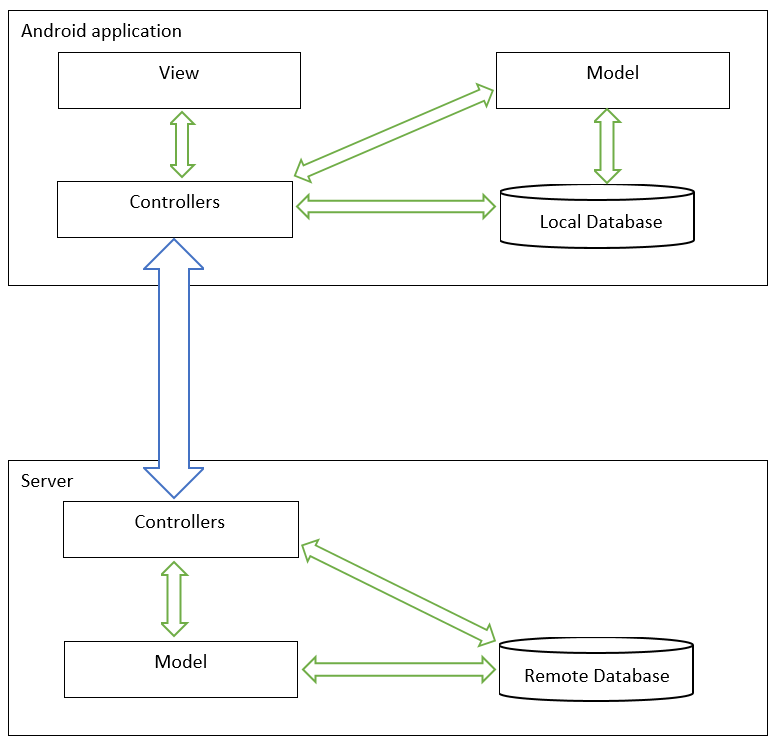
[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. The MVC is the main architecture of the system. Therefore, the system is decomposed into three main packages called model, view and controller.

The view package contains the classes of Android views (Activities) , controller package contains the classes of controllers or handlers which handle the data and instruction passing between the model classes and view classes. The model package contains the model classes which handles the mapping of data stored in the databases.

## Overview

The overall decomposition of the design model in terms of its package hierarchy and layers is as follows.



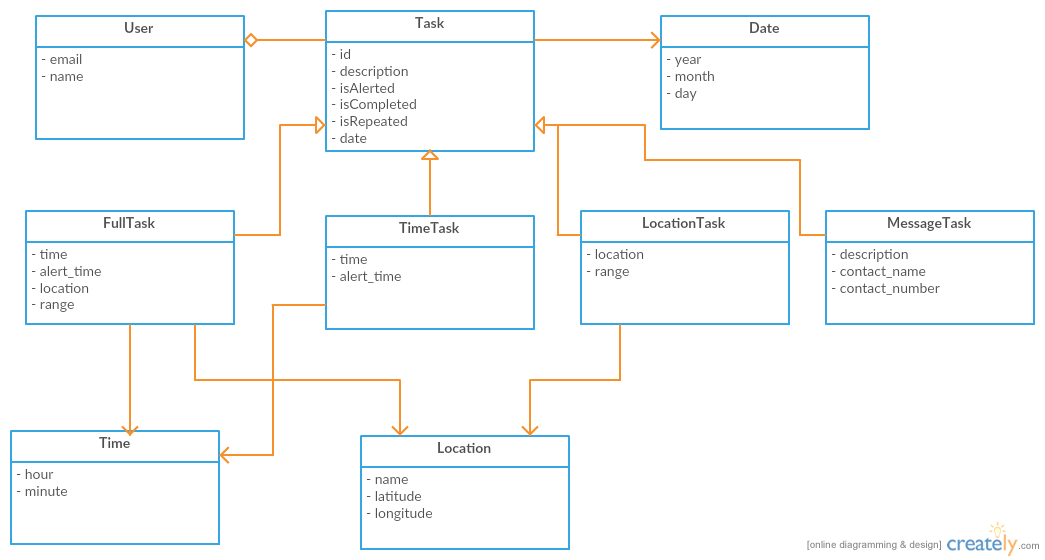
## Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

**Include the Class diagram and describe it**

The structure of the system is shown in terms of classes and their relationships. The aggregation and inheritance relationships are identified. The important attributes of the classes are identified to present a complete logical design view of the system. The class diagram is as follows.



# Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

**Include the Activity diagram and Sequence diagram and describe**

This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). The section is organized by groups of processes that communicate or interact.

## Activity Diagrams

|  |  |
| --- | --- |
| Register the user | |
| Application user | System |
| Request Sign in  Select a google account | Create a user account in the server  Sign in with google  View google sign in options  Send email address to server |

|  |  |
| --- | --- |
| Create a new task | |
| Application user | System |
| Request To provide task details  Submit task details | Show the view to enter details  Save the task details  Schedule the task |

|  |  |
| --- | --- |
| Check for predicted schedule | |
| Application user | System |
| Confirm the schedule  Select the tasks in the schedule  Request predicted schedule  Submit the date | Generate the predicted schedule  Send the request to the server  Schedule the tasks  Save the tasks in the schedule  View the schedule  Send the schedule to the application  Ask to enter the date |

|  |  |
| --- | --- |
| Notify a task | |
| Application user | System |
| Select task status  Yes  No  Is task Complete? | Save the task as incomplete  Save the task as completed  View a notification  Retrieve task details from Database  Alarm fires |

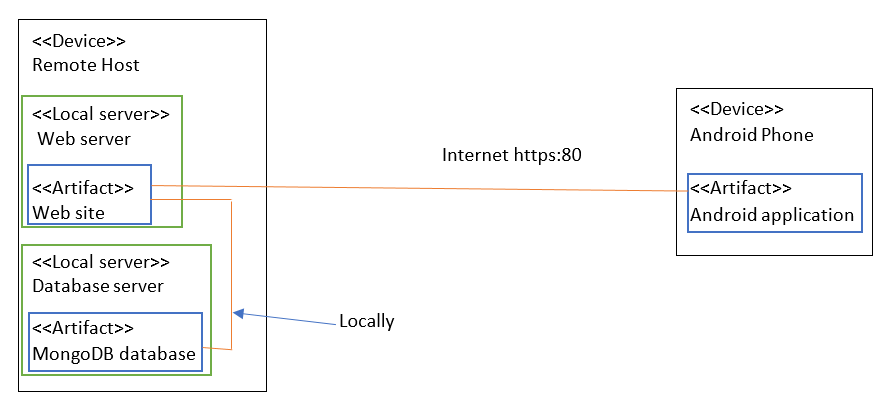
|  |  |
| --- | --- |
| Analyze daily data | |
| Application user | System |
|  | Generate reports  Update predicted schedule tables  Retrieve data from database |

## Sequence diagrams

# Deployment View

[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]Include the Deployment diagram if available and describe

The “Smart Planner” system includes two physical nodes. One is the remote host which host the web server and the remote database server. The interaction between the database server and the web server will be local to the remote host. The other node is the Android device (a phone) which holds the Android application.



# Implementation View

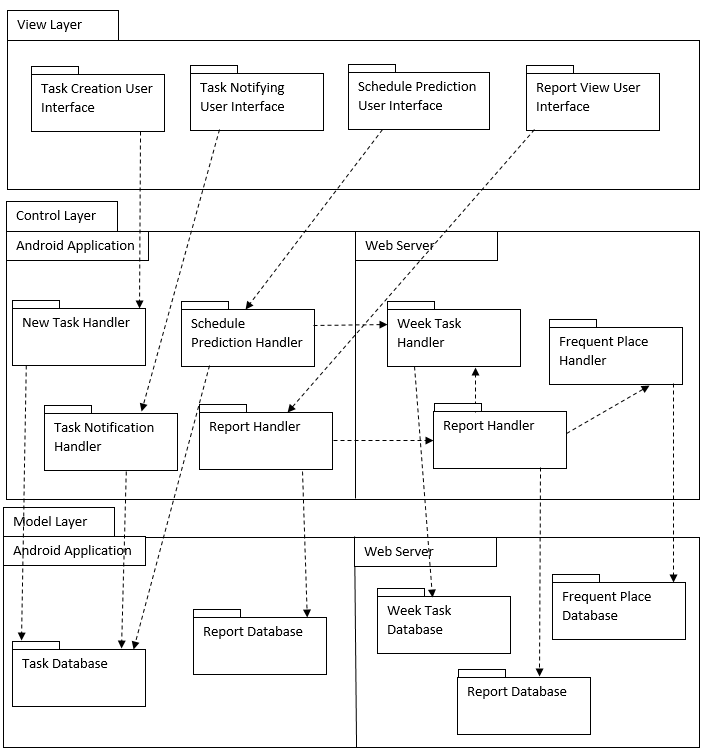
[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and architecturally significant components. The MVC architecture is followed within the Android application. The Android application and the web server follows the client-server architecture model.

## Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

The system is divided into three layers following the MVC architecture. The model layer, the view layer and the control layer are the main layers. The following diagram shows the major components within each layer to with their interactions.



## Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

**Include the Package diagram and describe**

* Model layer

Components within this layer handles the persistent data within the system. The data stored in the database are mapped to the model classes before accessing them. Further the data handling part of the system is done by the layer.

* View layer

This layer consists of the user interfaces of the system. The user interaction handling is done via the components of this layer.

* Control layer

This layer connects the model layer and the view layer. This layer responds to the user inputs and perform relevant interactions with the data model. Pre-data processing is done within this layer before transferring data to the data models. Client-server architecture components are included within this layer. (client control components and server control components)

# Data View

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

The task related data should be locally stored in a SQLITE database in the phone. But the most of the data of every user should be stored within a MongoDB (a NoSQL database) database. The data model should efficiently handle the data stored. A relational data modelling technique will be used within the web server.

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

* Responsive UI elements

The Android application should provide a responsive UI to make the user engaged to the system. Simple and descriptive UI elements should be used to achieve that.

* Accuracy of the schedule prediction

With time, more and more data will be collected. Therefore, the accuracy of the schedule prediction should increase as well.

* Time taken to generate a predicted schedule

The time taken by the server to retrieve past data and generate a predicted schedule should be minimum as possible. Or else the performance of the system will decrease.

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]

* Extensibility

The Android application must be easily extensible to add more functionality to the system. The data stored in the remote database should be easily varied due to the use of a NoSQL database.

* Reliability

The Android application must notify the scheduled time-based tasks accurately on time. Location-based tasks reliability will be somewhat less due to the unreliability of GPS functionality.

* Understandability

The user interfaces should be able to easily understood and used by an any user with a little knowledge of English language. The images and icons used should clearly depict the meaning of them.

**12. References**

Refer any data/ information in a standard format (eg. IEEE referencing style)

For different algorithms/ techniques/ theories you can refer text books.

For tools you can refer web pages.

For similar work you can refer research paper articles that describe the work.

You may include white paper articles for the description of technologies; web URL for the tool references. When you refer such a web page, you have to indicate the (Accessed on <<date>>)