Software Architecture Document

for

Virtual Room Reservation Assistant

Version 2.0 approved

Prepared by Group 26
B10715044 Kuo-Yen Chang
B10715029 Yen-Wei, Chen
B10730226 Wen-Yen, Tseng
B10715024 Yin-Cheng, Wang

Submitted in partial fulfillment
Of the requirements of
CS3025301 Software Engineering

2021/01/08

Table of Contents

Software Architecture Document

1.Introduction

1.1 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.

1.2 Scope

The Software Architecture Document provides an architectural overview of the VRRA System. The VRRA system is developed to support and maintain the conference rooms whether available or not.

1.3 Definitions, Acronyms, and Abbriviations

Acronyms and Abbriviations	Definitions
VRRA	Virtual Room Reservation Assistant

1.4 Reference

https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm

2.Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model developed using Rational Rose.

3.Architectural Goals and Constraints

There are some key requirements and system constraints which are significant on the architecture. They are:

1. The existing legacy of conference rooms which are registered in VRRA system must be accessed to retrieve all information. The VRRA system must support the data formats and DBMS of the legacy conference rooms.
2. All registered people can access the system from both PCs and remote PCs with the Internet.
3. The VRRA system must ensure complete protection of data from unauthorized access. All remote accesses are subject to user identification and password control.
4. The VRRA system will be implemented as a client-server system. The client portion resides to the end users and the server portion will temporarily operate on developers PCs in development version.

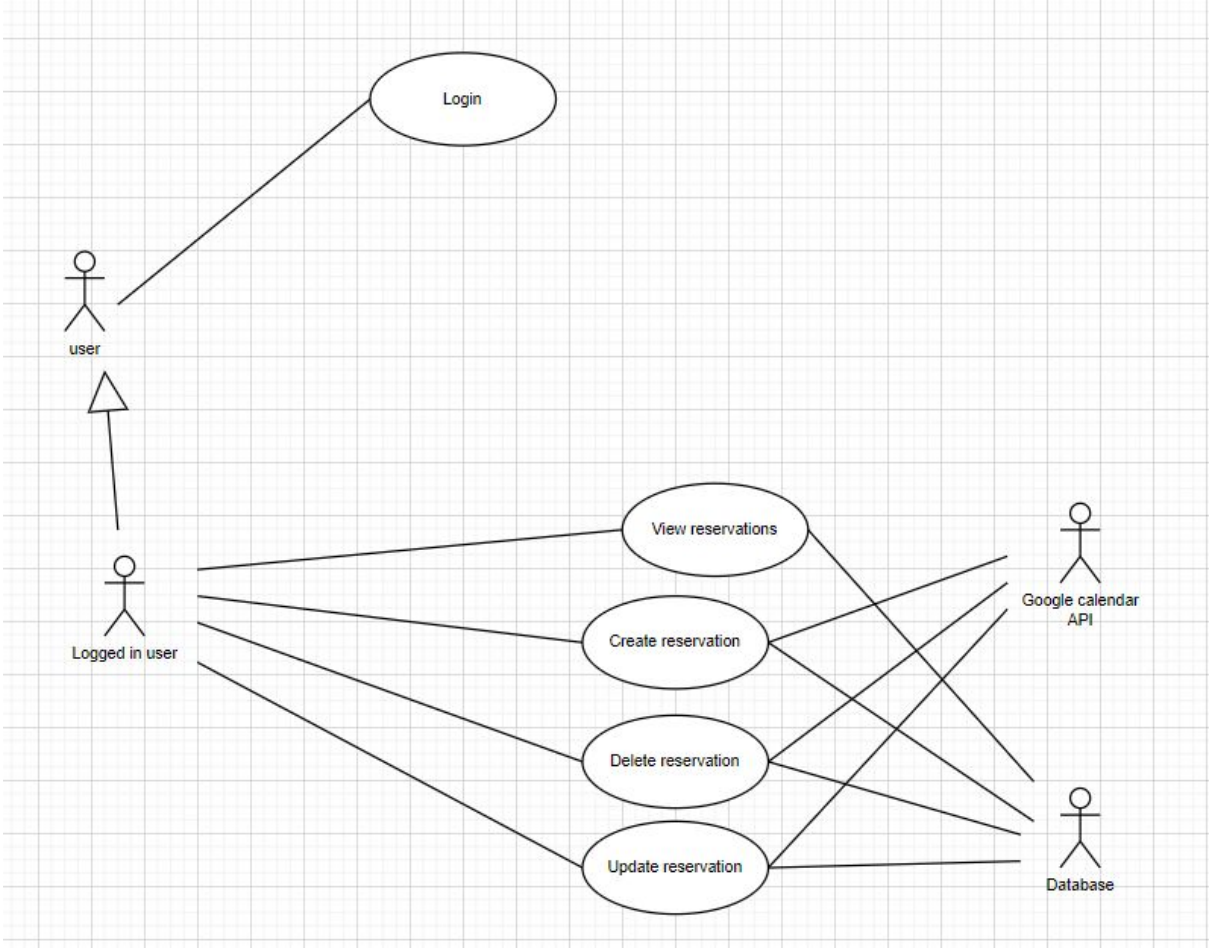
4.Use-Case View

A description of the use-case view of the software architecture. The use-case view is important for inputs to the selections of the set of scenarios and/or use-cases focus on an iteration. It describes the central, significant functionality of scenarios, and also describes the set of scenarios and/or use-cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The VRRA use-cases are:

- View reservations
- Login
- Create reservation
- Delete reservation
- Update reservation

4.1 Architecturally-Significant Use Cases



4.1.1 View reservations

Brief Description: This state allows a person views the availability of rooms.

4.1.2 Login

Brief Description: This state describes how a user logs into a VRRA system.

4.1.3 Create reservation

Brief Description: This state allows a user to create a reservation. This includes the size of the room, the location of the conference, and the time when the conference will go on.

4.1.4 Delete reservation

Brief Description: This state allows a user to delete the reservation for any kind of issue.

4.1.5 Update reservation

Brief Description: This state allows a user to change and update his/her options. This includes all the options from 4.1.7, but the user must beware of the time as when the conference is coming up soon (ex: before one day).

5.Logical View

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

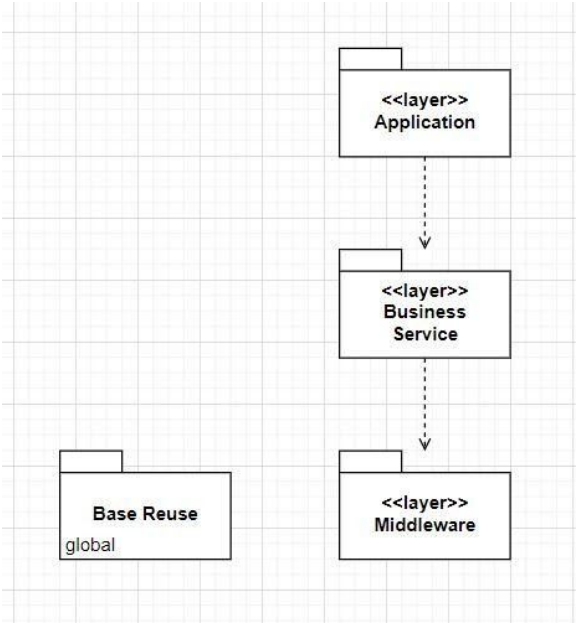
The logical view of the Virtual Room Reservation Assistant system comprises the 3 main packages: User Interface, Business Services, and Business Objects.

The User Interface Package contains classes for each of the forms that the actors use to communicate with the System. Boundary classes exist to support login, maintaining date, maintaining host info, maintaining reservation, selecting room, submitting notification, maintaining participants' info, and viewing reservation.

The Business Services Package contains control classes for interfacing with the reservation system, controlling host registration, and managing the reservation.

The Business Objects Package includes entity classes for the reservation artifacts (i.e. room offering, schedule).

5.1 Architecture Overview - Package and Subsystem Layering



5.1.1 Application layer

This application layer has all the boundary classes that represent the application screens that the user sees. This layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

5.1.2 Business Services layer

The Business Services process layer has all the controller classes that represent the use case managers that drive the application behavior. This layer represents the client-to-mid-tier border. The Business Services layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

5.1.3 Middleware layer

The Middleware layer supports access to Relational DBMS and OODBMS.

5.1.4 Base Reuse

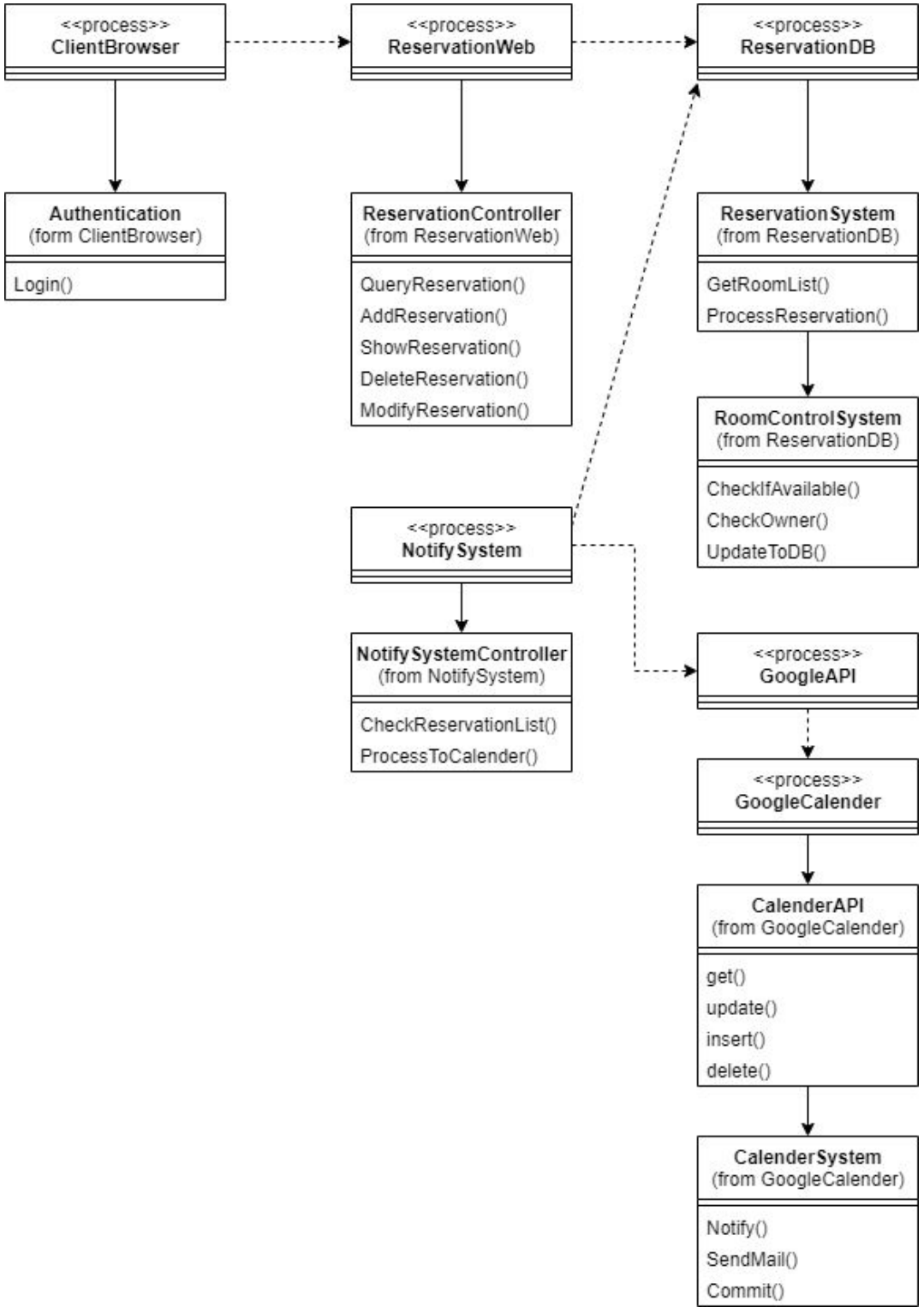
The Base Reuse package includes classes to support list functions and patterns.

6.Process View

A description of the process view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

The Process Model illustrates the room reservation classes organized as executable processes. Processes exist to support client registration, host functions,room reservation, meeting notification.

6.1 Processes



6.1.1 ClientBrowser

This is the Web page that directs the user to access the ReservationWeb.

6.1.2 Authentication

Manage the user account and access the system.

6.1.3 ReservationWeb

Manages the reservation functionality, including user interface processing and coordination with the business processes.

There is one instance of this process for each user that is currently reserved for room.

6.1.4 ReservationController

This supports the use case allowing a user to reserve for room in the current session. The user can also modify or delete room reservation if changes are made within the add/drop period at the beginning of the session.

Analysis Mechanisms:

- Distribution

6.1.5 ReservationDB

This is the database that records the all status of reservation.

6.1.6 ReservationSystem

This is the reservation system that controls the reservation process.

6.1.7 RoomControlSystem

This is the room control system that allocates the room.

6.1.8 NotifySystem

This is the notification system that controls the notification process.

6.1.9 NotifySystemController

This supports the use case allowing a system to send the notification to upcoming reservations to the owner and invited user via Google Calendar.

6.1.10 GoogleAPI

This is the API that is provided by google in order to access the Google Calendar.

6.1.11 GoogleCalender

This is the system that Notify System uses to send the notification to the user.

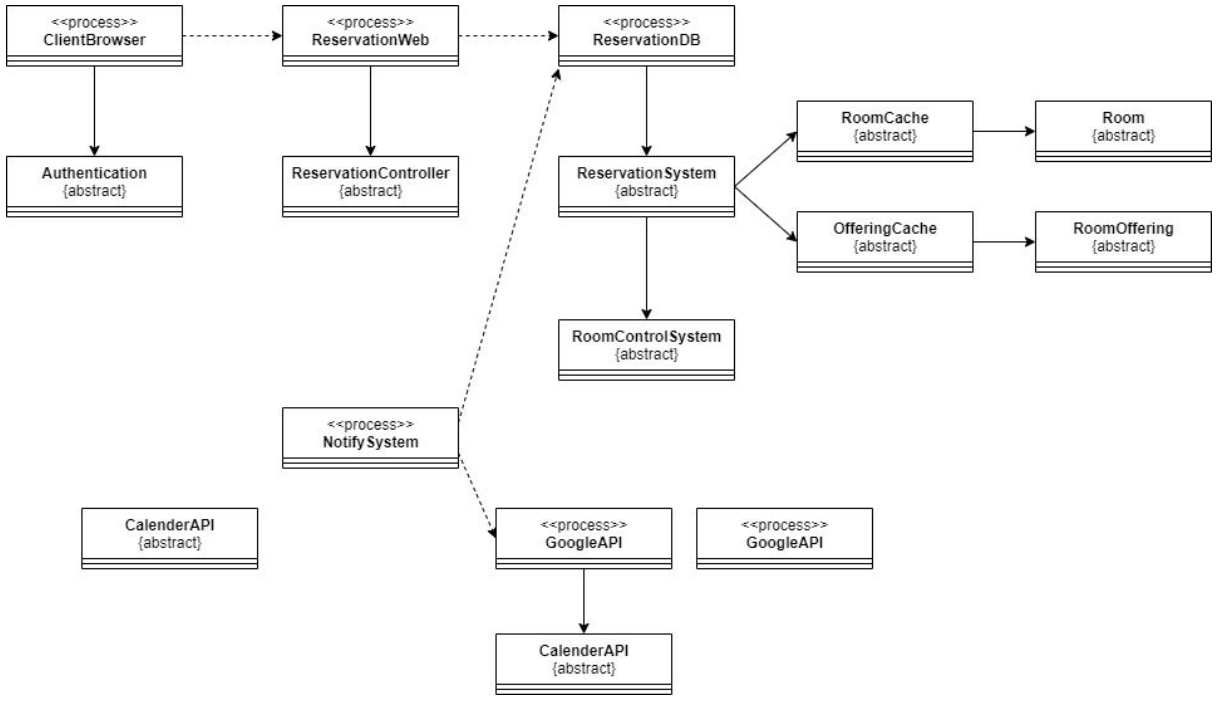
6.1.12 CalenderAPI

This is the API that Google provides in order to send the notification by Notify System.

6.1.13 CalenderSystem

The system that notify users and says your reservation is coming.

6.2 Process to Design Element



6.2.1 RoomCache

The Room Cache thread is used to asynchronously retrieve items from the legacy Room Catalog System.

6.2.2 OfferingCache

The Offering Cache thread is used to asynchronously retrieve items from the legacy Course Catalog System.

6.2.3 Room

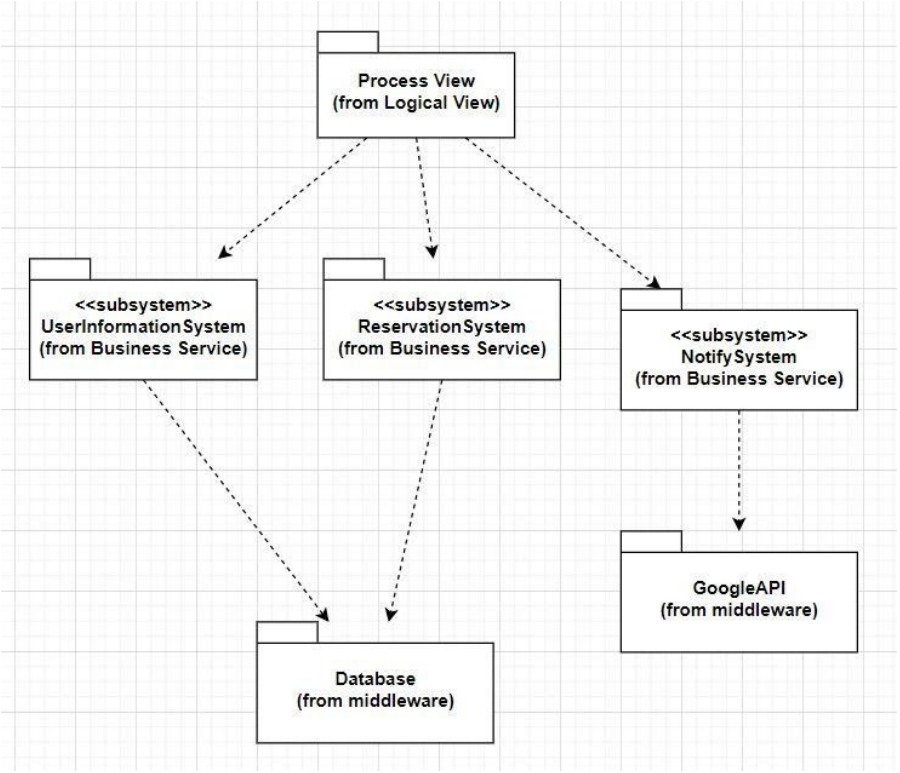
The conference rooms are offered by any location from the country.
Analysis Mechanisms:

- Persistency
- Legacy Interface

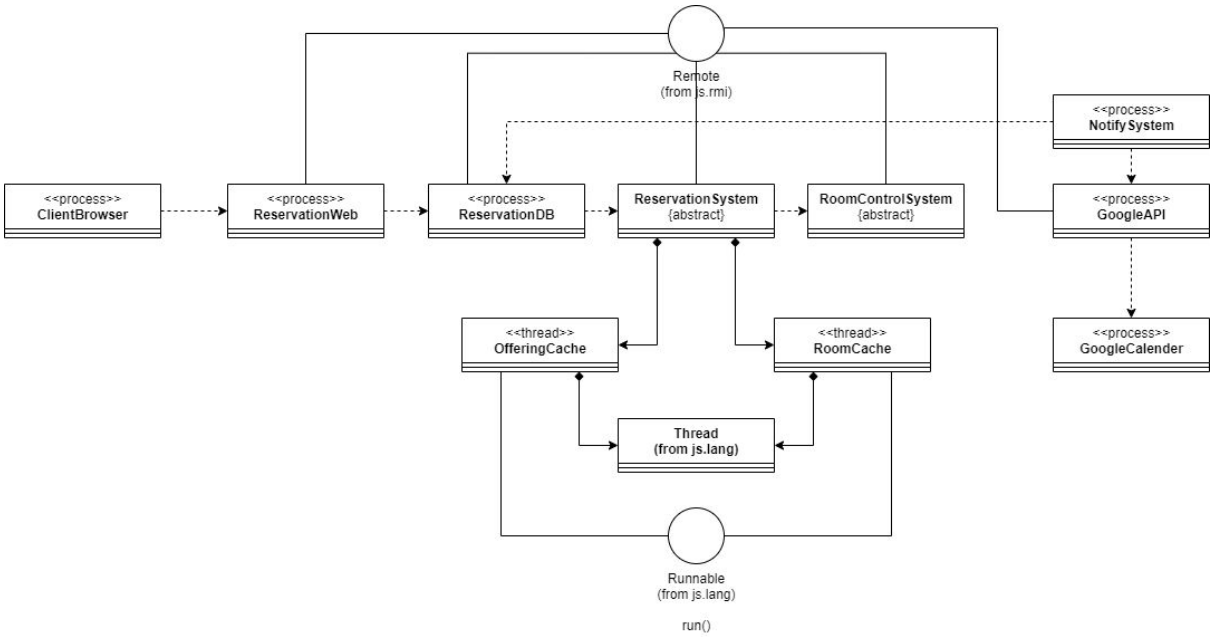
6.2.4 RoomOffering

A specific offering for a room, including spare days, room size, and participates.
Analysis Mechanisms:
- Persistency
- Legacy Interface

6.3 Processes Model to Design Model Dependency



6.4 Processes to the Implementation



6.4.1 Remote

The Remote interface serves to identify all remote objects. Any object that is a remote object must directly or indirectly implement this interface. Only those methods specified in a remote interface are available remotely.

Implementation classes can implement any number of remote interfaces and can extend other remote implementation classes.

6.4.2 Runnable

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. The class must define a method of no arguments called run.

This interface is designed to provide a common protocol for objects that wish to execute code while they are active. For example, Runnable is implemented by class Thread.

Being active simply means that a thread has been started and has not yet been stopped.

6.4.3 Thread

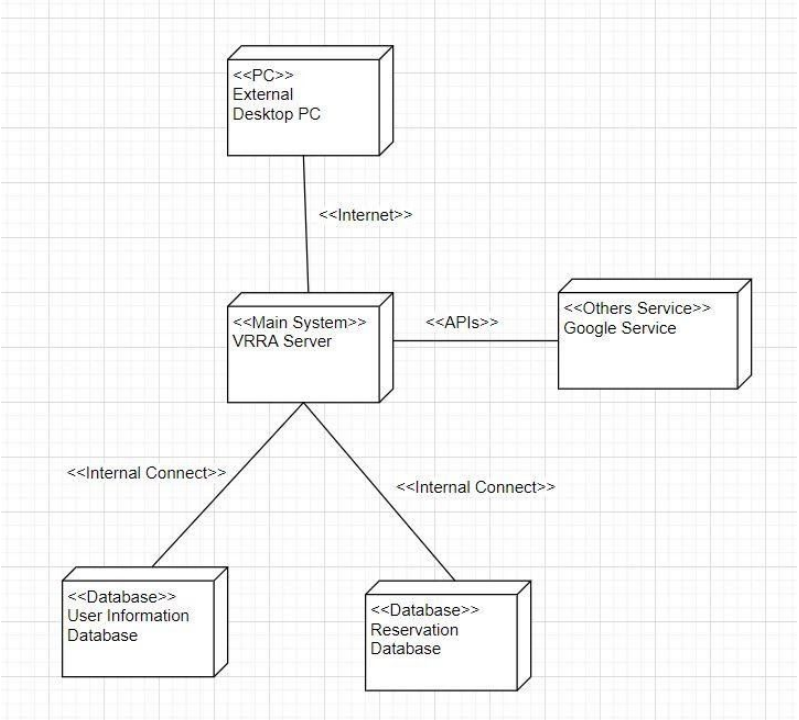
A thread is a thread of execution in a program. JavaScript allows an application to have multiple threads of execution running concurrently.

Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.

7.Deployment View

A description of the deployment view of the architecture describes the various physical nodes for the most typical platform configurations.

This section is organized by the relation which connect between each system or physical equipment; each such configuration is illustrated by a deployment diagram, followed by a mapping of processes to each processor.



7.1 External Desktop PC

Users access our service by using their own desktop PC which is connected to VRRA server via Internet.

7.2 VRRA Server

The server runs our services' backend. We will deploy the VRRA system on it.

7.3 Google Service

Including Google Calendar and GMail services. VRRA server will use Google APIs to connect these outside services.

7.4 User Information Database

The database stores the user's personal information, including account names, passwords, email addresses, tokens, and so on. VRRA service will use some internal connection method like program's function call or other to operate these datas.

7.5 Reservation Database

The database stores the data about reservation information, including host name, participants' names, participants' emails, date, and so on. VRRA service will use some internal connection method like program's function call or other to operate these datas.

8.Size and Performance

The system shall support up to 2000 connections and 500 simultaneous operations.
The system must be able to complete 99% of all transactions within 1 minutes.
The system must respond in 1 minute..
The system must show current status if the system is busy. (EX: Loading… / Please wait… / etc.)
Though the client accesses this system via browser. But the download data for the browser must not exceed 200 MB disk space and 256 MB RAM.

9.Quality

The software architecture supports the quality requirements as follow:

1. The user shall have the browser, and requires Firefox 83, Chrome 86, Edge 85, Safari 84 or higher. Also, javascript shall be enabled.
2. The user interface shall be designed ease-of-use for arbitrary computer users with no additional training.
3. The VRRA system shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time except the server platform has some problem.
4. After new updates released, users can automatically get updates by reconnecting to VRRA system