**Software Architecture Document**

**Dream Destinations Getaways**

Produced by:

Erika Valle-Baird

Madison Bilko

Collin Cavanaugh

Tamara Greer

Mohammed Khoshkhoiyazdi

Domeniko Kodra

Table of Contents

[Introduction 2](#_Toc108347498)

[Architectural Representation 2](#_Toc108347499)

[Context View 2](#_Toc108347500)

[Use-Case View 2](#_Toc108347501)

[Logical View 3](#_Toc108347502)

[Process View 3](#_Toc108347503)

[Deployment View 3](#_Toc108347504)

[Implementation View 3](#_Toc108347505)

[Architectural Goals and Constraints 3](#_Toc108347506)

[Context View 4](#_Toc108347507)

[Use Cases 5](#_Toc108347508)

[Use-Cases Significant to the Architecture 6](#_Toc108347509)

[Significant Use-Case Descriptions: 11](#_Toc108347510)

[Logical View 13](#_Toc108347511)

[Logical View Diagram 14](#_Toc108347512)

[Presentation Package Diagram 15](#_Toc108347513)

[Application Package Diagram 16](#_Toc108347514)

[Domain Package Diagram 17](#_Toc108347515)

[Content Package Diagram 18](#_Toc108347516)

[Profile Package Diagram 19](#_Toc108347517)

[Support Package Diagram- 20](#_Toc108347518)

[Persistence Package Diagram- 20](#_Toc108347519)

[Process View 20](#_Toc108347520)

[Deployment View 23](#_Toc108347521)

[Implementation View 24](#_Toc108347522)

[Size and Performance 24](#_Toc108347523)

[Quality 24](#_Toc108347524)

# Introduction

**Purpose**

The purpose of the Software Architecture Document is to provide an overall view of the structure of the system. This is done by using different architectural views to show different modules and components of the system. The important architectural decisions that are made regarding the system are shown within this document. This document is used for both developers and non-developers who make up a team so that decisions regarding the system can be made based on how the system is designed without having to go into the code to see what is happening internally.

**Scope**

The Software Architecture Document applies to the overall structure of the software. It is used to show developers and non-developers what goes on within the system without going into the code. It can help influence decisions that are to be made by a team regarding architectural decisions that need to be made about the project. For example, if a module or component within the system is not desirable, then a team who is working on the system may use this document to see how other things are interacting and then decide based on their findings.

# Architectural Representation

Presented in this document is the architecture of our system viewed from several different perspectives. The perspectives presented include the Context view, Use-Case view, Logical view, Process view, Deployment view, and Implementation view.

# Context View

**Audience**: all stakeholders of the system, including acquires, users, and developers.

**Area**: The Context View graphically depicts relationships and interactions between the system and external entities.

**Related** **Artifacts**: The Context Model

# 

# Use-Case View

**Audience**: all the stakeholders of the system, including the end users.

**Area**: The Use-Case View graphically depicts scenarios and/or use cases substantial and essential functionality of the system.

**Related** **Artifacts**: Use-Case Model

# Logical View

**Audience**: Designers and programmers.

**Area**: Functional requirements that describe the design’s object model coupled with the description of the most significant use-case realizations.

**Related** **Artifacts**: Design Model, Package Diagram.

# Process View

**Audience**: Integrators and programmers.

**Area**: The Process View describes the non-functional requirements of the system including the design’s concurrency and synchronization aspects.

**Related** **Artifacts**: Process View Diagram

# Deployment View

**Audience**: Deployment managers and system administrators.

**Area**: The Deployment View describes the mapping of the software onto the hardware and shows the system’s distributed aspects.

**Related** **Artifacts**: UML Deployment Diagram

# Implementation View

**Audience**: Programmers

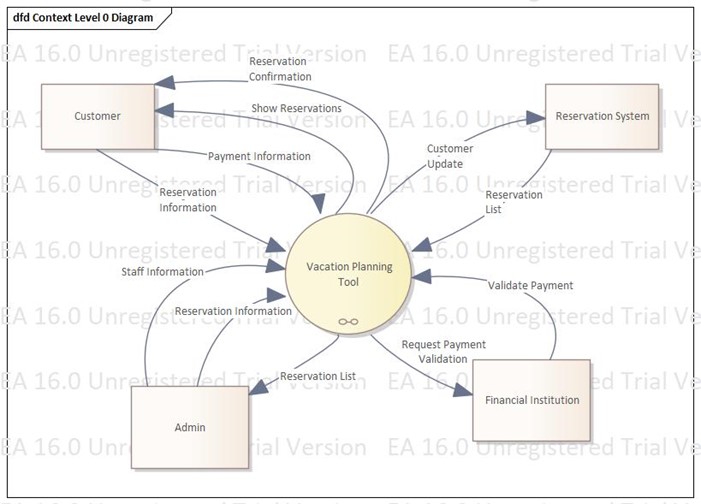
**Area**: The Implementation View describe both subsystems and layers of the application known as software components

**Related** **Artifacts**: Components, Implementation Model

# Architectural Goals and Constraints

The software requirements and objectives that have some significant impact on the architecture include security, privacy, distribution, and reuse. Security and privacy are important because the user’s data must be secure and protected from other people who may want to steal this data. Distribution is important because this software must be available to whoever wants to use it without fail. Reuse is important because users must be able to reuse this software as desired without issue. These features impact the architecture since they must be present and able to work with the other features that are present within the architecture. The design and implementation strategy, development tools, team structure, and schedule are impactful as well. The design and implementation strategy can make or break the system. Without proper design and implementation, the architecture can become messy and disorganized, which may lead to a system that does not work properly, if at all. The development tools are also important, as improper tools can also lead to a system not working as intended. Without a proper team structure, faulty decisions may be made about the architecture. This can lead to problems within the system and further extend the time needed to finish the project. This ties into how the project schedule is an important factor. If the schedule is too tight, then things may get missed within the system and the architecture may not be up to par. If more time is given within a schedule, then more time can be used to ensure the architecture is good and to make any decisions about changes that may be needed.

# Context View



**System Externals**

**Customer**

A customer is any user of the system who is not an employee and who does not have an admin account. A customer may search for flights, car rentals, hotels, and packages via reservation. A customer may submit payment information to complete a reservation order. A customer who provides departure and return dates in a flight query will receive the results of such a query. A customer who provides valid payment information will receive a rese

rvation confirmation upon completion.

**Admin**

An admin is any user of the system who is an employee and who holds an admin account. An admin is informed about reservation information. The admin may update employee data and reservations.

**Reservation System**

The reservation system is a database that contains customer and reservation information. The reservation system is informed about customer updates so that it can provide reservation lists specific to the customer’s need.

**Financial Institution**

An institution responsible for all financial transactions acting as a sales channel. The financial institution is informed about any payment requests and returns responses to those requests.

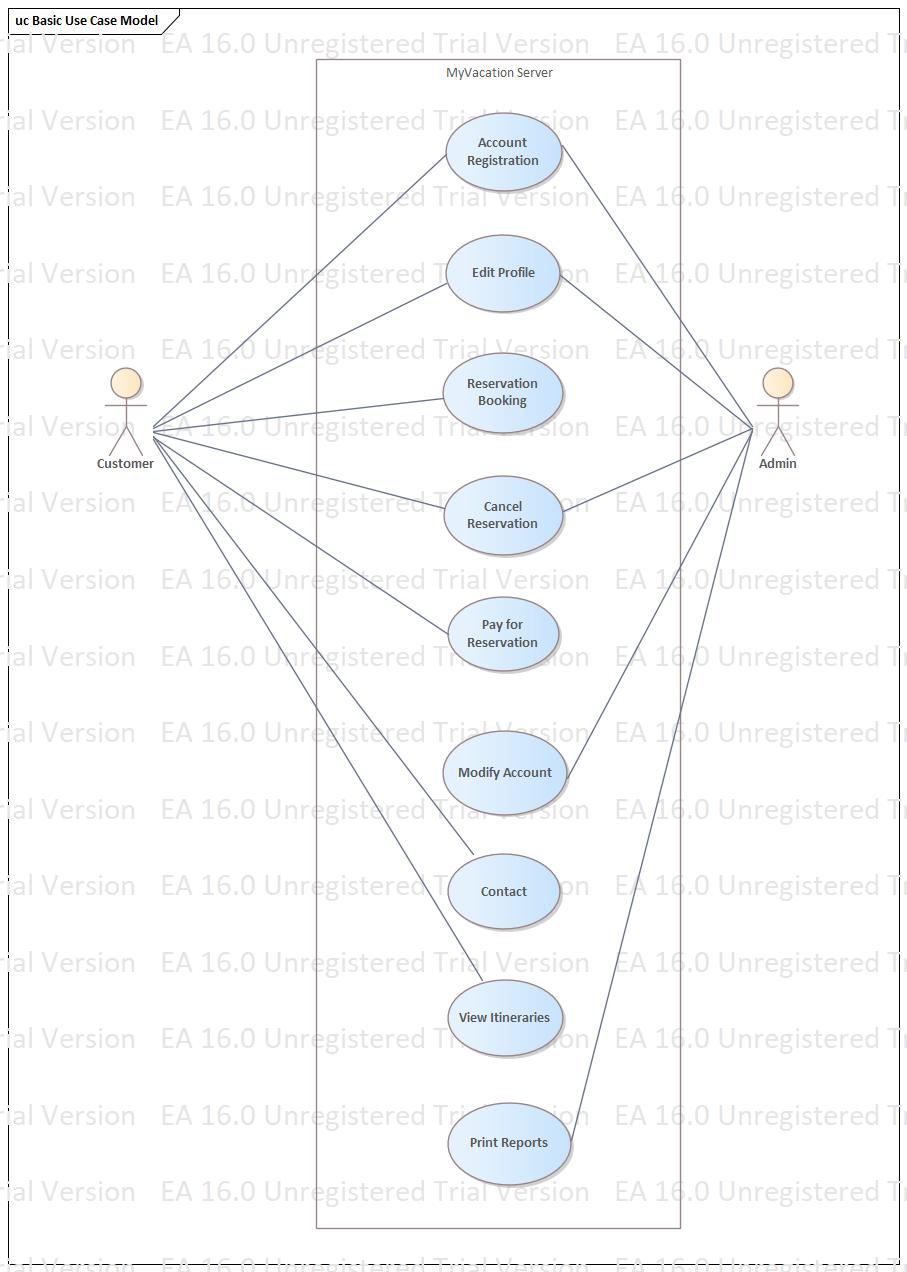
# Use Cases

The Use-Case view describes the set of scenarios and use cases critical to the central functionality of the architecture. Users of this system are actors who include both customers and administration personnel. These users interact with the systems interface using a web portal. Information requests from such users are sent to the MyVacation server where data is stored and retrieved.

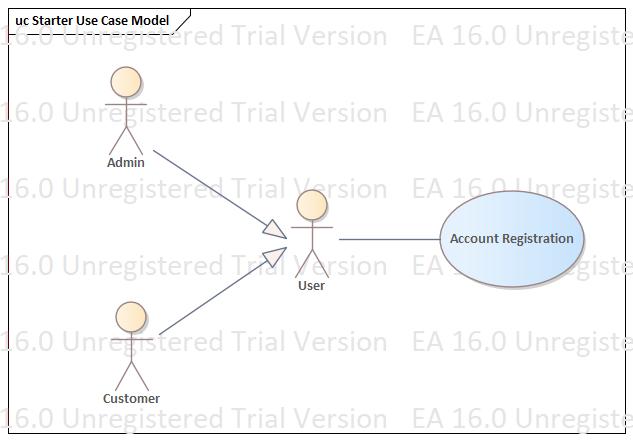
## Use-Cases Significant to the Architecture

* Subscribe
* Edit Profile
* Book Flight
* Book Package/Hotel/Car Rental
* Cancel Reservation
* Pay for Reservation
* Print Reports
* Add/Delete/Modify User’s Account
* Contact Company
* View Itinerary
* Print Itinerary

**Figure 1 – Overall Use Cases**



**Figure 2 – Potential Account Registration Use Cases**

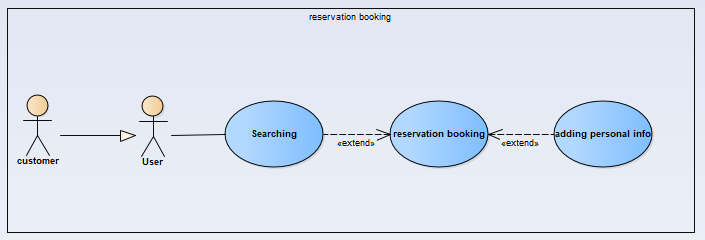


**Figure 3– Potential Edit Profile Use Cases**

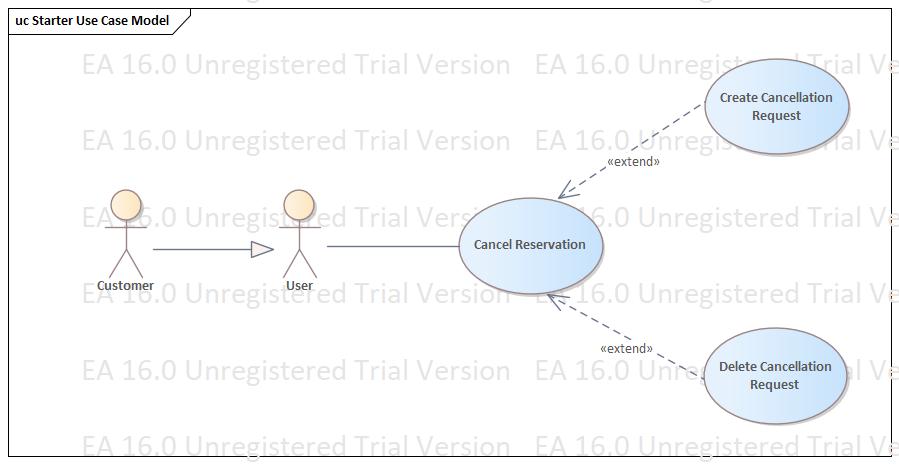
Diagram

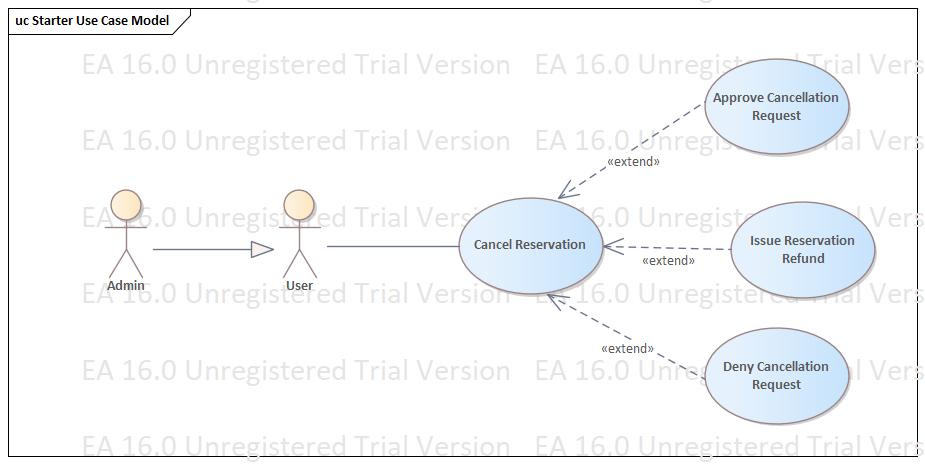
Description automatically generated

**Figure 4 – Potential Reservation Booking Use Cases**



**Figure 5 – Potential Cancel Reservation Use Cases**





**Figure 6 – Potential Pay for Reservation Use Cases**

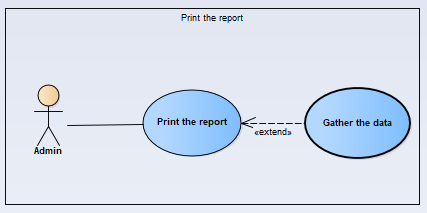
A picture containing diagram

Description automatically generated

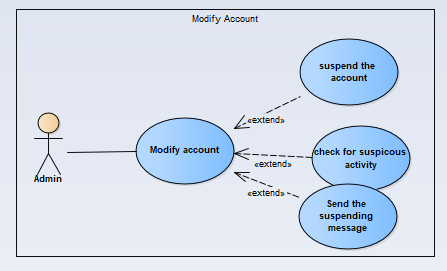
Diagram

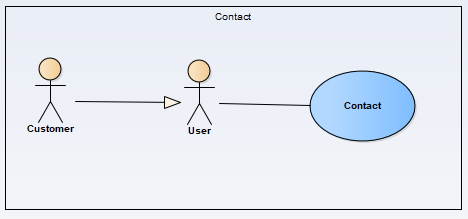
Description automatically generated

**Figure 7 – Potential Print Reports Use Cases**

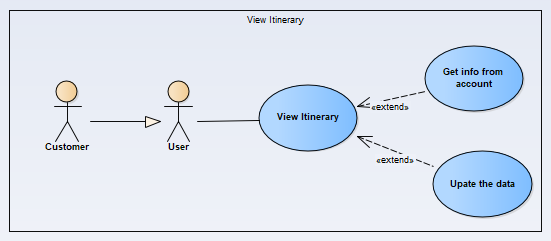


**Figure 8 – Potential Modify Account Use Cases**



**Figure 9 – Potential Contact Company Use Cases** 

**Figure 10 – Potential View Itinerary Use Cases**



## Significant Use-Case Descriptions:

1. Account Registration

This use case occurs when a potential account user wants to register with our service. This includes retrieval of user information and sending said information to the MyVacation system.

1. Edit Profile

This use case occurs when an account holder desires to alter their user profile. The user can change personal information such as name, home address, email address, card on file, password, and phone number.

1. Reservation Booking

This use case occurs when a user decides on a particular flight to a vacation destination. The user will then add what seat and food they may enjoy on the flight. The booking will also include how many people are included and what additional package amenities may be involved. Once the purchase is completed, the reservation will be updated in the MyVacation’s server and will be under the user’s account history.

1. Cancel Reservation

This use case occurs when a customer submits a cancellation request. This submission will notify the administration, who then may approve or deny the request. If a cancellation request is approved, a refund request is sent to the third-party card processing and the MyVacation server data is updated.

1. Pay for Reservation

This use case occurs when the user has finished customizing their flight package with various amenities and proceeds to the checkout process. The user will then use a card on the account or add a different card to pay for the flight. The card information will be checked on to ensure accuracy and then process the payment if the card is valid. A receipt will be shown and emailed to the user. The MyVacation server will be updated with this new data.

1. Print Reports

This use case occurs when the administration wants to have the data reports printed. The system will retrieve the data and check for any updates, along with obtaining the newest data. The system then prints out a screen report screen for the administration and allows them to print out the report.

1. Modify Account

If a user account is locked by the MyVacation system due to suspicious behavior by the user, an administrator will review the account to ensure the user is behaving appropriately. Once the account has been reviewed, the administrator will either send a suspension message to the user’s about the suspension or unlock the account.

1. Contact Company

This use case occurs when the user wants to contact the company for assistance or any general questions they have about the website and the products and services.

1. View Itinerary

This use case occurs when the user wants to see the final booking information. The system will retrieve all the upcoming user’s travel information from the account and check for any updates. Then it will print out the information to the screen for user.

# Logical View

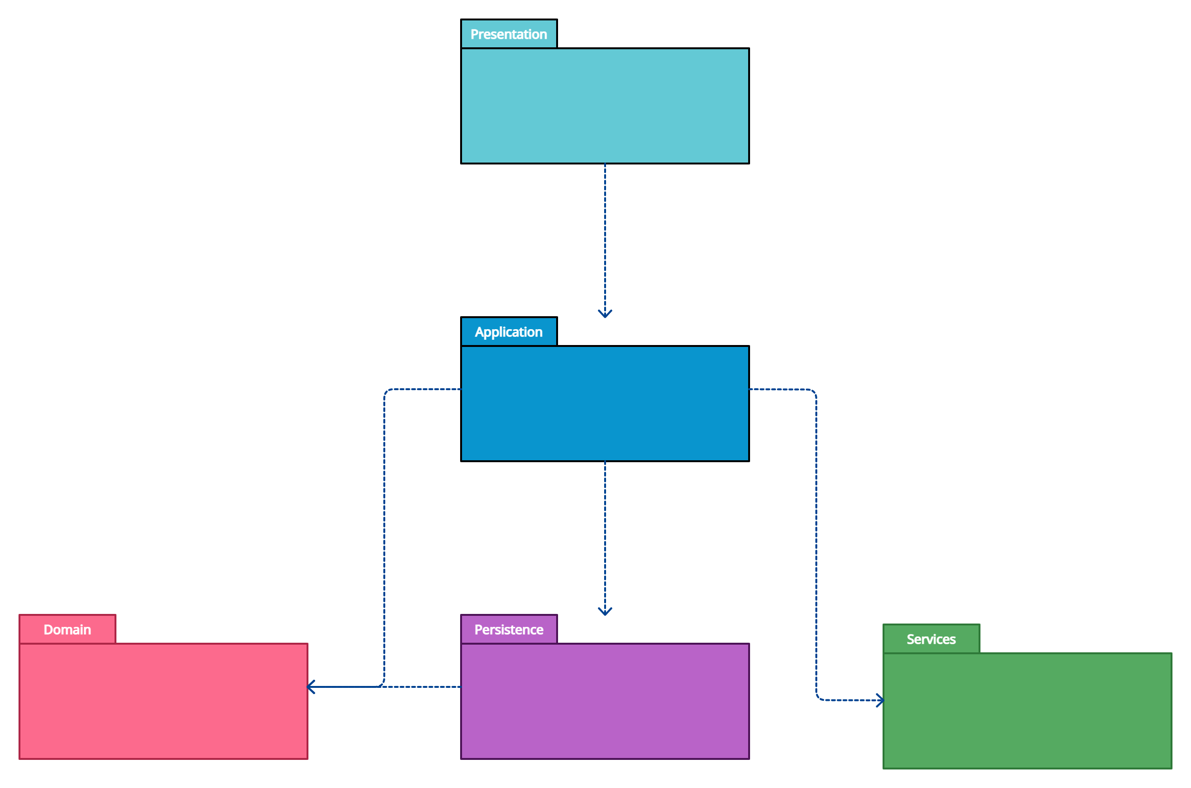
**Overview**

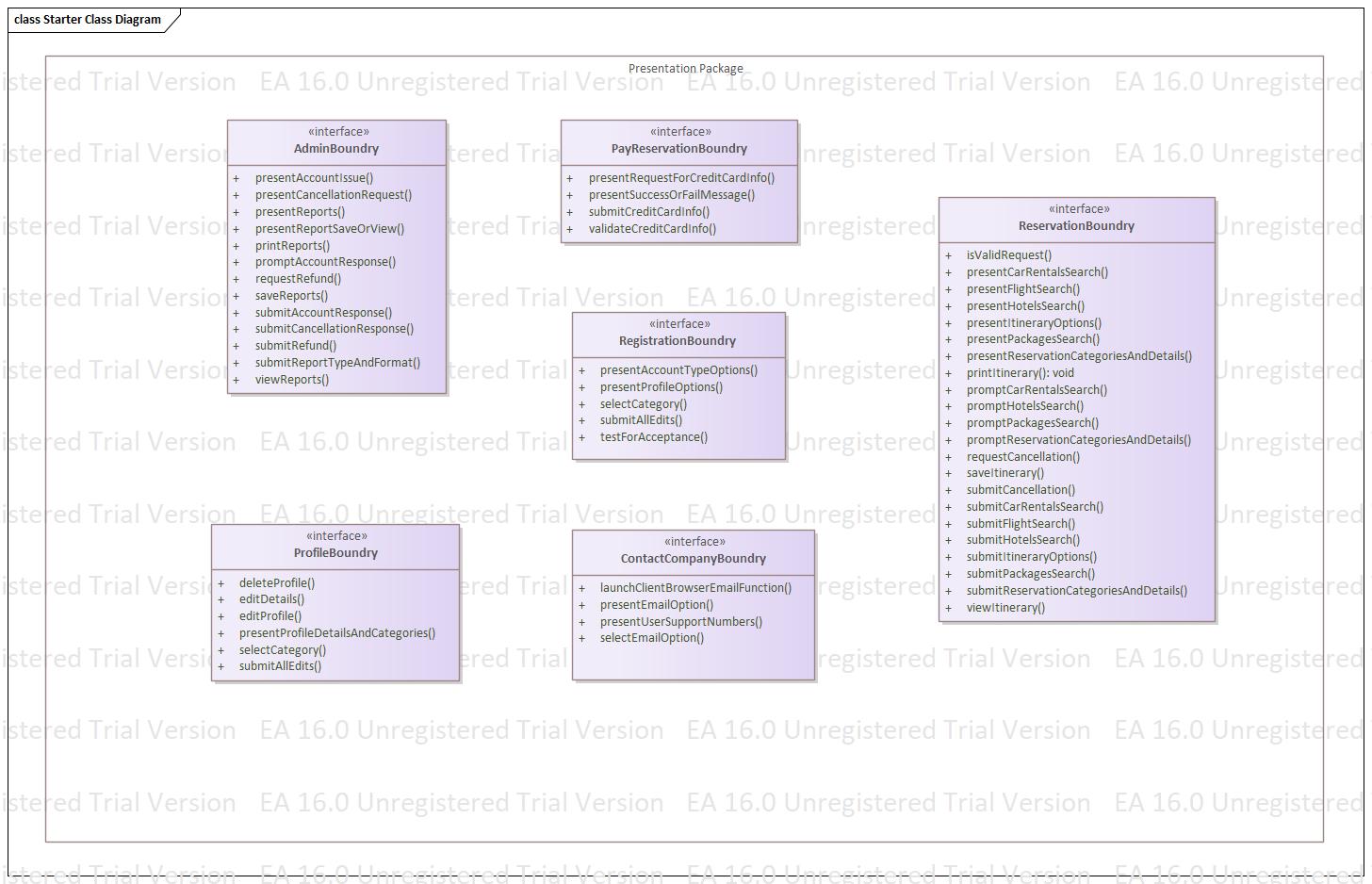
The logical view of the architecture describes important classes, their organization within the packages, and their organization within layers. Here, the functional requirements of the system are specified via a blueprint of the architecturally indispensable parts.

The logical view of the MyVacation Tool System is comprised of 5 main packages:

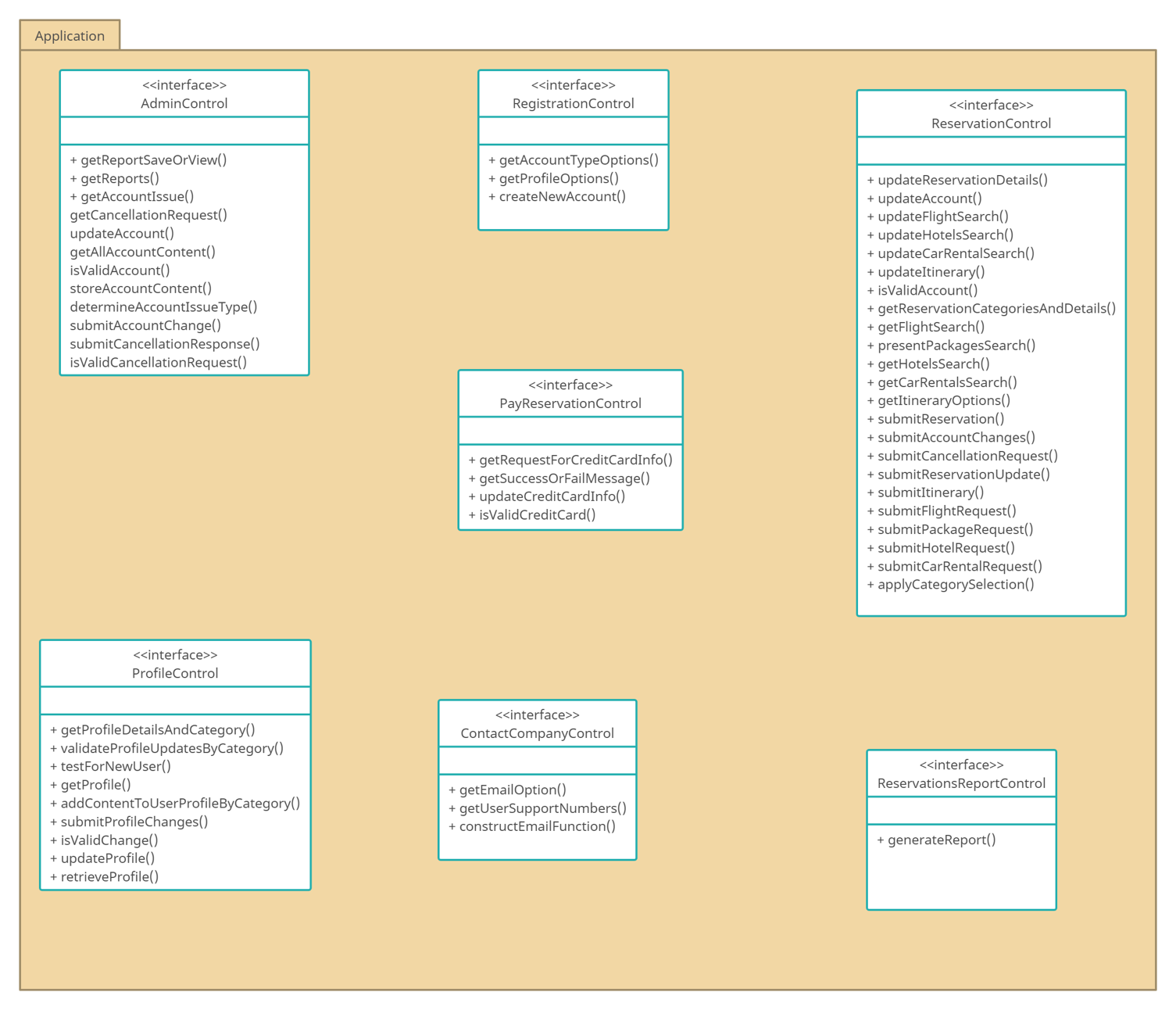
* **Presentation**
  + The presentation package includes the user interface, containing boundary classes for each of the different forms for actors interacting with the system. This package includes the interface that the user sees and interacts with when they use our system and page rendering. Boundary classes exist to support subscribing to our site, maintaining of profiles, reservation booking, reservation cancellation, purchasing a reservation, itinerary viewing and printing, print reports, and contacting the company.
* **Application**
  + The application package is comprised of classes that represent major processing functionality within the system. These classes are control classes that support content management, profile management, reservation processes, and reservation payment with credit card.
* **Domain**
  + Comprised within the domain package are classes that support profiles, content, and support.
* **Persistence**
  + The persistence package contains classes with objects that remain persistent in the system such as profiles.
* **Services**
  + To periodically perform database maintenance, the services package contains system-level classes that provide maintenance services.

### Logical View Diagram

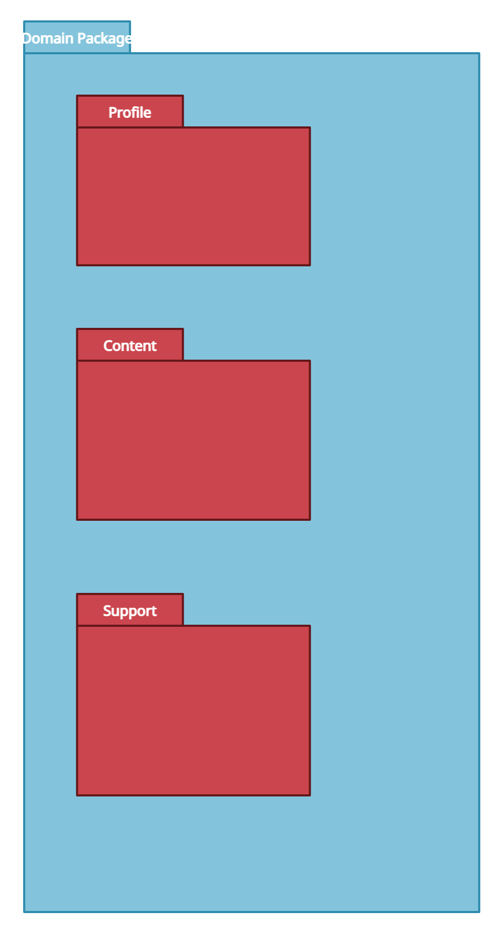


Presentation Package Diagram 

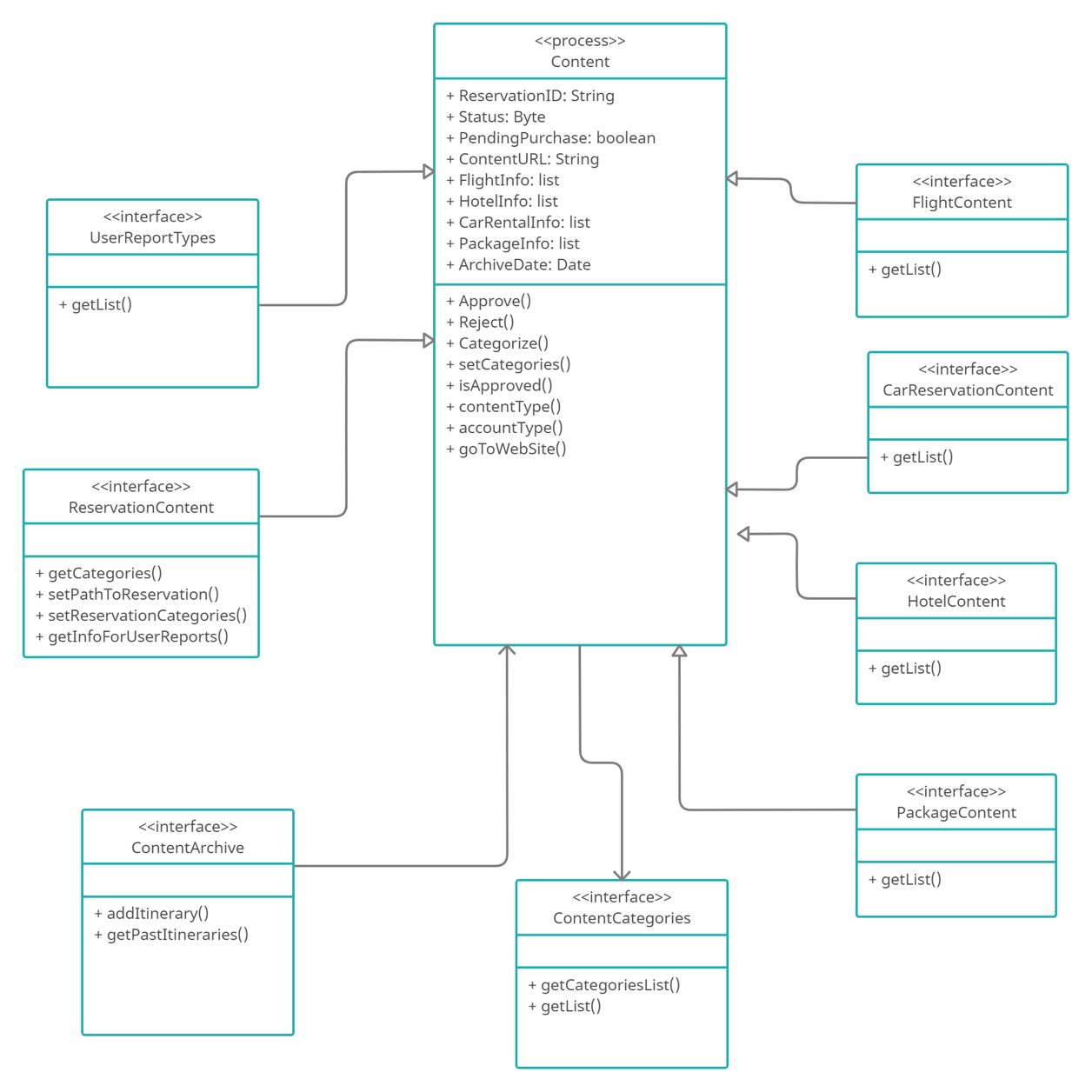
### Application Package Diagram



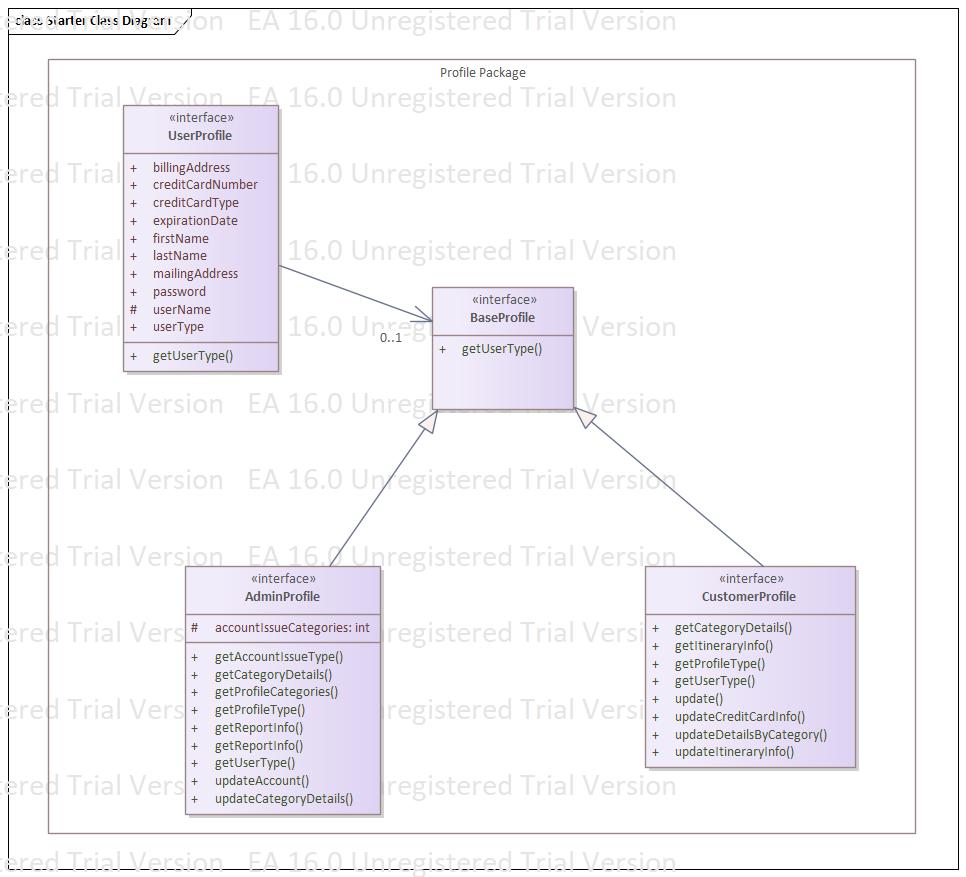
### Domain Package Diagram



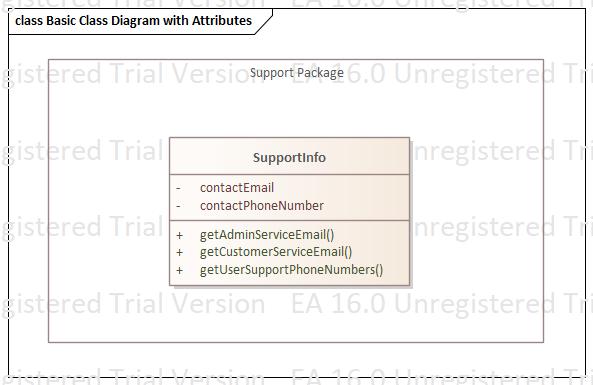
### Content Package Diagram



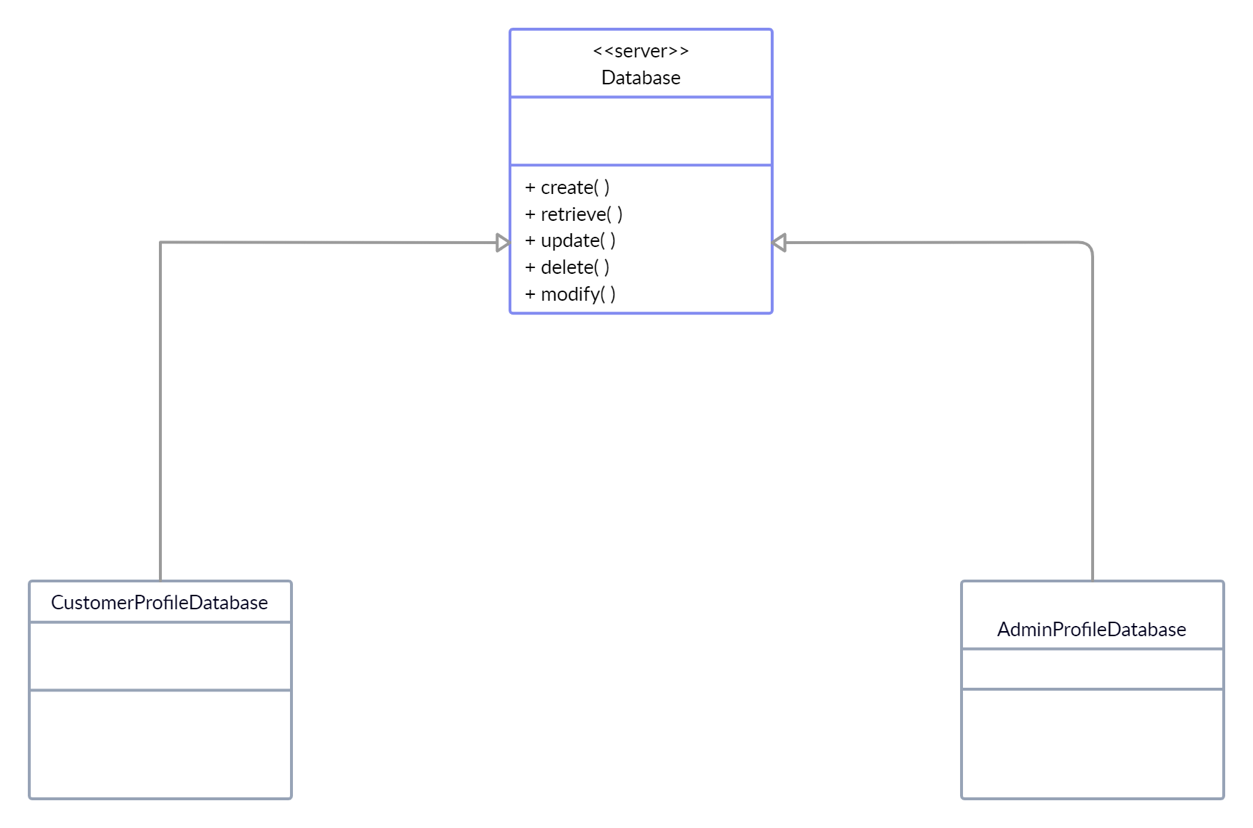
### Profile Package Diagram



### Support Package Diagram-

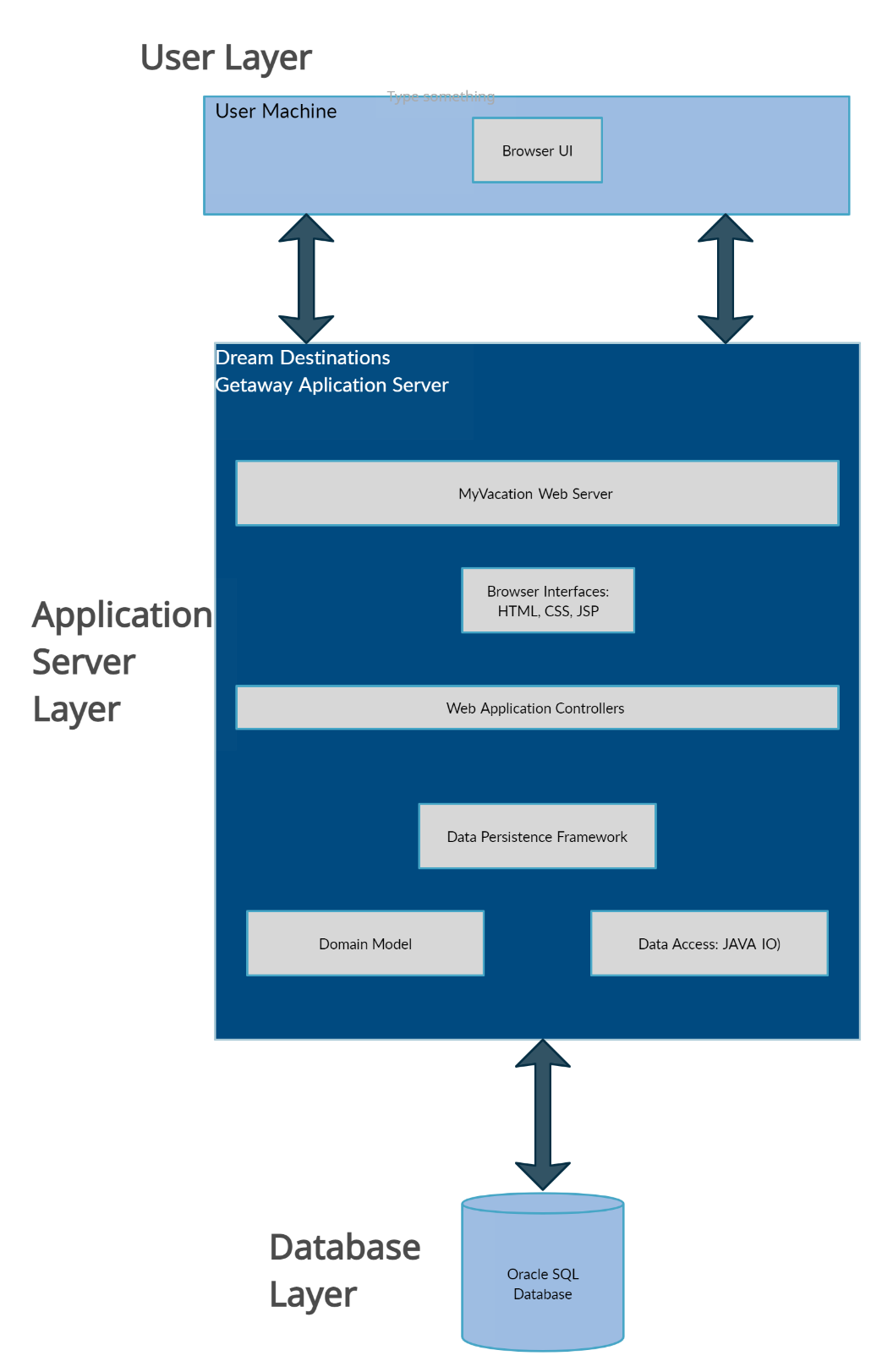


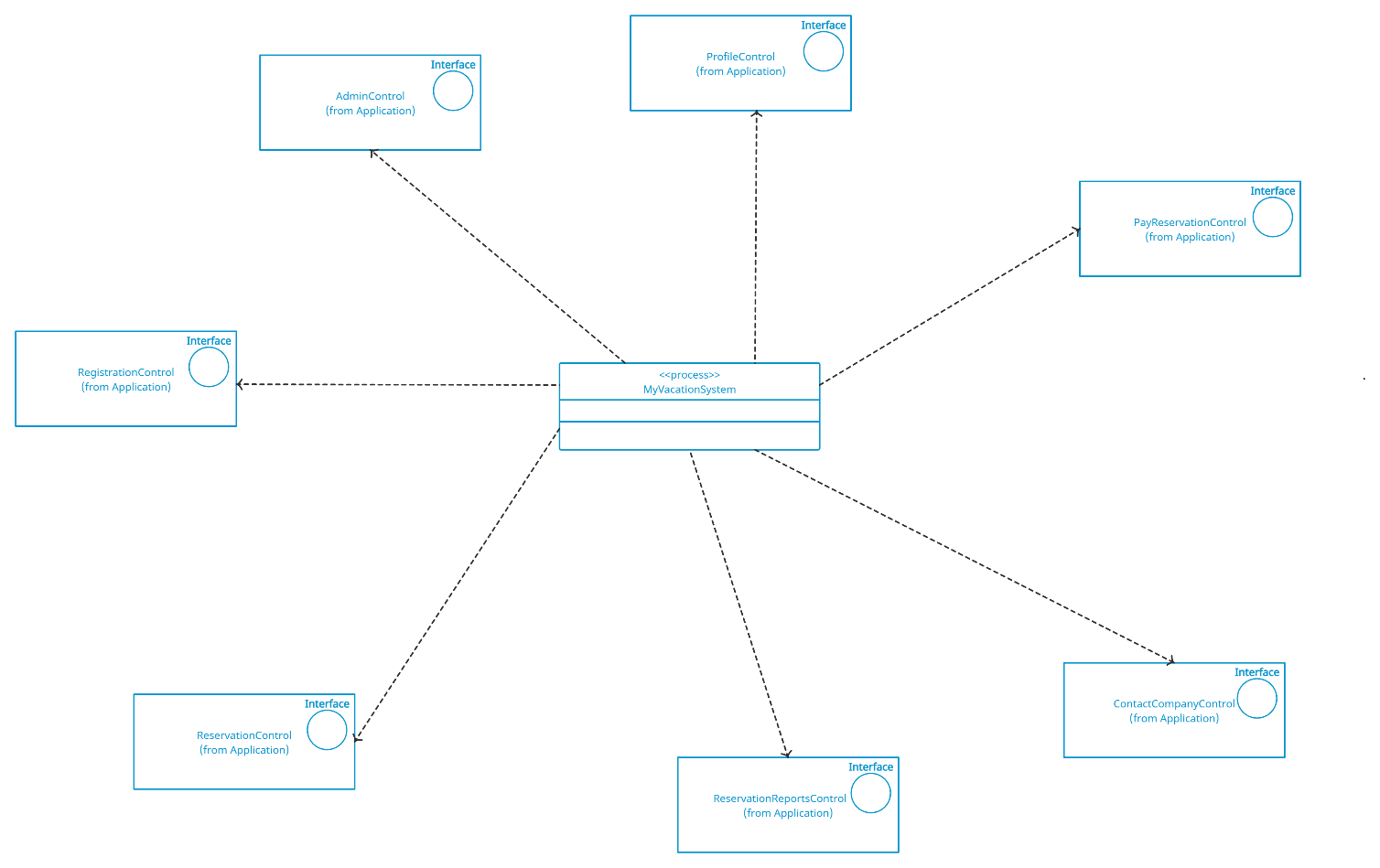
### Persistence Package Diagram-



# Process View

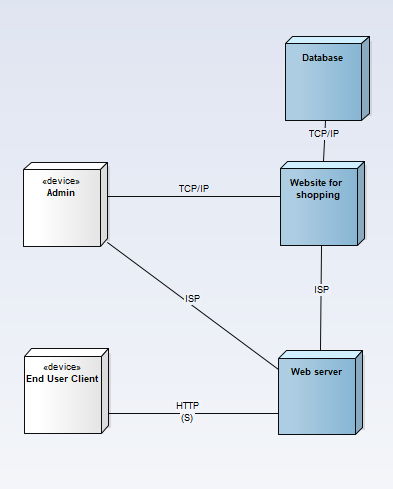
Contained in the process view is a description of the architecture from the processes and threads perspective. This is a description of the system’s decomposition into lightweight processes. A graphic illustration depicting the tasks that are involved in system execution, interactions, messages, and configurations. This model illustrates the classes that interact with the reservation process. Processes exist here to support account registration, profile, admin functions, access to the third-party external billing system, reservation search system, and content. Due to our small statue, currently only a single process will be required to provide server-level functions for the MyVacation System. A significant part of the process includes threads from the application functions, which handle the data submitted by the user at the presentation layer. The user interacts with our service by sending and receiving messages that move through here.





# Deployment View

The deployment view displays the network configurations for MyVacation. The user accesses the website using an HTTP connection to access the webserver through the Internet. The user can access this using any device that is capable of running a web browser. The web server communicates to the admin device and websites for shopping, using ISP connections. The website connects to any administration devices and databases using TCP/IP technologies.



# Implementation View

The Client accesses the web server through the first layer, this is all the users can interact with. The webserver can communicate with communicate with the second layer admin and the websites where the services are purchased. These websites are all drawing information from a database on a different level.

# Size and Performance

The architecture is impacted by the size of the program. The bigger the size of the program, the more complicated the architecture will be. Normally a bigger program means more features, which then results in the architecture having more modules and components. In terms of performance, the architecture shows how the system performs specific actions during specific time periods. This can include the time spent on an action and the number of actions that happen during a specific period.

# Quality

The software architecture contributes to capabilities such as maintainability, extensibility, reliability, and efficiency. The architecture contributes to maintainability because it can show what areas of the system are potentially not working as well as they could together, therefore providing insight as to what can be fixed. It can also show what parts of the system may need to be enhanced, which ties into extensibility. The architecture contributes to extensibility by showing what can be expanded and built upon effectively and efficiently. It can be used to show what modules and components interact well with each other and can grow to be more. The architecture contributes to reliability by showing what parts of the system may need work to improve reliability and what parts of the system are built well enough to be reliable. This may help speed up the process of troubleshooting what parts of the system are giving trouble with reliability. The architecture contributes to efficiency by showing what areas of the system do not work as efficiently as they should, which makes it easier to pinpoint what needs to be improved within the code.