Hotel Management System

-

Object Design

Table of Contents

<HMS>

OBJECT DESIGN DOCUMENT

Object Design Document (ODD) describes object **design trade-offs** made by developers, **guidelines** they followed for subsystem interfaces, the **decomposition of subsystems** into p**ackage**s and classes, and the class interfaces. The ODD is **used** to exchange interface information among teams and **as a reference during testing**. The audience for the ODD includes system architects (i.e., the developers who participate in the system design), developers who implement each subsystem, and testers.

Among three approaches to generate ODD, we follow "**ODD embedded into source code**" approach in SE301, since the other methods create many redundancies, inconsistencies.

The initial version of the ODD can be written soon after the subsystem decomposition is stable. Both packages and class interfaces can be generated from source code (comments!) by using a tool, which is named Javadoc. Keeping material for the ODD with the source code enables the developers to maintain consistency more easily and rapidly.

1. Introduction

1.1. Object Design Trade-offs

For the trade-offs:

- **Memory space vs. Response time:**
  Since our project HMS is focused on making the booking experience for the guests, and fast but convenient data input for the users, as well as clear and easy to understand display of data, that is why our system prioritizes Response time over Memory space.

- **buy vs. build:**
  In normal circumstances, we would have a study to weigh the advantages and disadvantages of having to build or buy the HMS system, but in our current situation, We only have one option, which is to build the system in question.

For naming conventions:

We used the Snakecase naming convention (Words are delimited by an underscore) for naming the Views.

We used the Pascalcase naming convention (Words are delimited by capital letters) for naming variables in python files.

For boundary cases and exception handling mechanisms:

Django handles most of the edge cases related to data entry and data storage.

As for exceptions, our implemented logic takes care of them on the backend side of things.

1.2. Interface Documentation Guidelines

- Classes are named with singular nouns and underscore between two words for names that contain more than one word.
- Methods are named with verb phrases, fields, and parameters with noun phrases. This can be noticed in the Views.py Files in our project.

1.3. Definitions, Acronyms, and Abbreviations

ODD: Object Design Document
HMS: Hotel Management System

User: any person who uses the system.
Guest: any person with the intention to spend time in the hotel and use its services.
Employee: any person working at the hotel in any capacity.
Admin: administrator of the hotel also known as General manager.
Manager: manages staff and receptionists and deals with events hosted by the hotel, deals with guests in cases where he is needed.
Receptionist: deals with guest related matters and helps guests when necessary.
Staff: Can be cleaners, bellboys, waiters, cooks, and chefs.

GUI: A GUI or graphical user interface is a form of user interface that allows users to interact with electronic devices through a graphical interface.
HTML: Hyper Text Markup Language. It is the standard markup language for documents designed to be displayed in a web browser.
CSS: Cascading Style Sheets. CSS is a style sheet language used for describing the presentation of a document written in HTML.
UX: User Experience. UX abbreviation is used to define the design process to create products that provide meaningful and proper experiences to users.

1.4. References

Currently there is no system in place to be replaced by the system we are building.

<HMS>

2. **Packages**

File organization of the code:

- **Hotel_Management_System**
- **Group_documents :** includes mockups and er diagrams used to help I mplementation
  - **HMS:**
    - Accounts: includes the **views.py**
    - HMS: includes **urls.py** and **settings.py** file for django
    - Main_pages:
      - **Admin.py**: includes registrations for the models
      - **Forms**
      - **Models.py**: contains models for the database
      - **views.py**
    - Static
      - Css :css files
    - Templates: every sub package contains html pages related to said user.
      - Admin
      - Common_pages
      - Employee
      - Guest
      - Incs : includes pages used for html inheritance.
      - Manager
      - Receptionist
      - Staff
      - Index.html : the template used to fill with other pages using inheritance to avoid cod redundancy.
    - Manage.py: allows us to run the local server and make migrations.

<HMS>

3. Class Interfaces