SBI Managment system

Kevin Patel
Jeff Jordan)

Abstract

SBI Management System is a program for use in the commerciel Hospitality business in order to easily navigate the processes of booking, reserving, and maintaing rooms in an impactful way to alleviate streses of overbooking and non-rentalable rooms within businesses. So far the project has understood many of its final capabitlies whilst knowing what is needed out of and is soon to flush out the proper requirements to achieve those.

1. Introduction

The purpose of the SBI Management system is to remove the clerical errors of record keeping by hand and potential mistakes that could happen with note taking with roating cast of managerial staff and employees. Condensing and centralizing all reservation and booking information into one simple, easy-to-use application in order to remove the stress of record keeping and room tracking.

1.1. Subsection Heading Here

Occasionally you need to break your sections into separate parts, you will likely not need a subsection for every section

1.1.1. Subsubsection Heading Here. Occasionally you will need to break your subsections into separate parts, if you find yourself using this often, you're likely going overboard. Don't try to go any lower down than this...

1.2. Background

A member of our group's family owns a hotel. Owning a small business means that you don't have access to a corporations resources and systems. You're left to your own devices. This project is our response to that need. It is a practical idea for a practical business. The design will be centered around a simple GUI to quickly assign rooms and check availability.

Primarily, it will be used for the family's business, but we will design it to be implemented by any local small hotel or motel. Organizational needs should always be taken care of by software in this age.

1.3. Challenges

Neither group member has much experience working with GUI's. There's little doubt that the biggest hurdle will be integrating code into framework.

The second challenge will be understanding the framework's objects and how to manipulate them. It's almost like learning a completely new programming language.

Finally, the most constant hurdle will be time. We still haven't worked out an actionable schedule or plan yet. Our individual classwork responsibilities have taken precedence up to this point.

2. Scope

- Check-in guests
- Checkout guests

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

1	Release occupancy
2	Fill vacancy

TABLE 1. USE CASES

2.1.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.2. Use Cases

Use Case Number:	1
Use Case Name:	Booking and Maintaing Rooms
Description:	Administrators at motel and hotel businesses will be able to block rooms once they are booked by customers or repairs/cleaning needs to take place.
Use Case Number:	2
Use Case Name:	Reservations
Description:	Interact with a calender in order to properly plan out and schedule rooms to be booked and blocked to prevent overbooking.

You will then go on to (minimally) discuss a basic flow for the process(To Come):

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number:	2
Use Case Name:	Checkout
Description:	A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.