Software Requirements Specification Document for Urban Clap

Submitted to:

Ms. SIRAT KAUR

Submitted by:

ID	Name
1711981252	SAHIL GOEL
1711981268	SATWINDER SINGH
1711981272	SAURAV
1711981276	SHASHANK KUMAR

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the **Urban-Clap**, an android application. It will explain the purpose and features of the system, interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for the acquirer which is the owner in this case.

1.2 Scope

This platform is to make our urban lives more fulfilling to solve our needs in a clap. They want to be the go-to platform helping customers to complete the projects that are important to their lives. It enables users to find any service professional like

- plumber
- wedding photographer
- yoga teacher
- interior designer
- learning arts
- filing taxes
- getting healthier

1.3 Definitions, Acronyms, and Abbreviations.

Table 1

TERM	DEFINITON
Database	Collection of all the information monitored by this system
User	Any person or company logged on the system.
Android	A mobile device operating system developed by Google Inc
DFD	Data Flow Diagram.
MacOS	Macintosh Operating system that is now known as the classic MacOS.
Software Requirements	A document that completely describes all of the functions of a proposed system and

Specification (SRS)	the constraints under which it must operate. For example, this document.
MS-OS	Microsoft Operating System is an operating system by Microsoft.
GUI	GUI is the Graphical User Interface.
iOS	iOS is a mobile operating system created and developed by apple.
SQL	It is a structured query Language.

1.4 Overview

The upcoming chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the general factors that affect the product and its requirements and is used to establish a context for the technical requirements specification in the second chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product. This subsection of the SRS (as described in table1) will list each of the factors that affect the requirements stated in the SRS. Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

The fourth chapter Identifies the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements.

The fifth chapter Identifies the approvers of the SRS document. Approver name, signature, and date will be used.

The sixth chapter provides alternate ways to structure section 3 on the specific requirements. We will pick the best one of these to organize section 3 requirements.

The figures and diagrams are associated on the following pages:

Figure index 1

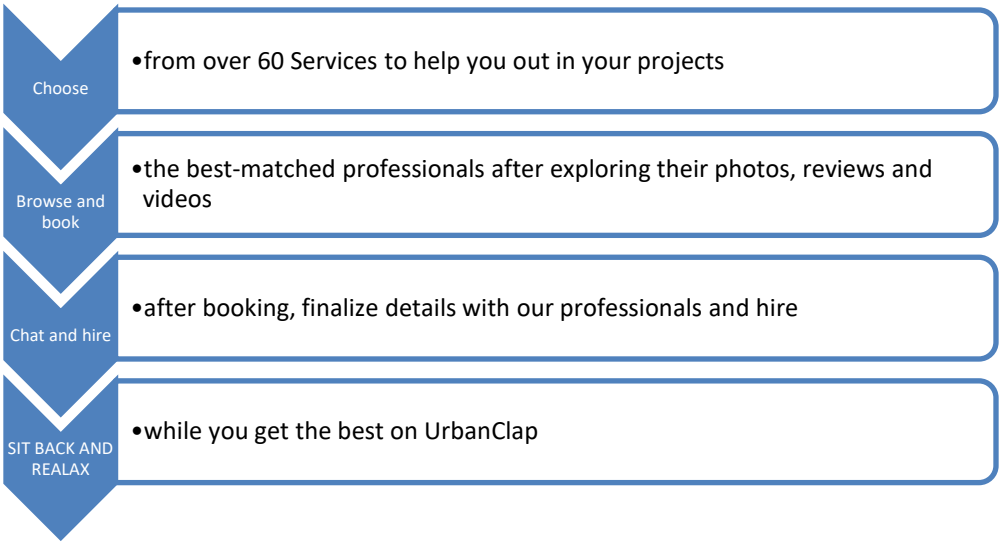
2 Overall description

UrbanClap Technologies India Pvt. Ltd. operates an online platform that helps customers to find and hire service professionals for their personal activities/needs.

2.1 Product Perspective

Urban Clap is a mobile application with a web service in order to find and hire professionals for their personal activities/needs. The mobile application will work on mobile Android/IOS devices. User will get a quotation for their job and then they can pick the accurate match. The history is stored and a user can view this from their login panel. This app is a hassle-free experience for getting your work done. The call & message history is also stored in this app's interface.

product perspective 2



2.1.1 System Interfaces

The software shall interface with SQL and Visual Basics.

2.1.2Interfaces

The software we are developing will interact with these main actors of the Urban Clap application-

- **Customer**: A customer can choose over the best-matched professionals after exploring their photos, reviews and videos.
- **Service-person:** The service person which will be best matched with the request of the customer will respond to the requests made by them.

2.1.3 Hardware Interfaces

The system has no hardware interface requirements.

2.1.4 Software Interfaces

This software is operable on MS-OS (As described in table1) and MacOS (As described in table1).

2.2 Product Functions

The product functions generally focus on the booking of the desired services with the best matched credentials.

2.3 User Characteristics

User should be familiar with the terms like login, register, order system etc.

2.4 Principle Actors:

Principle Actors are Customer and Administrator

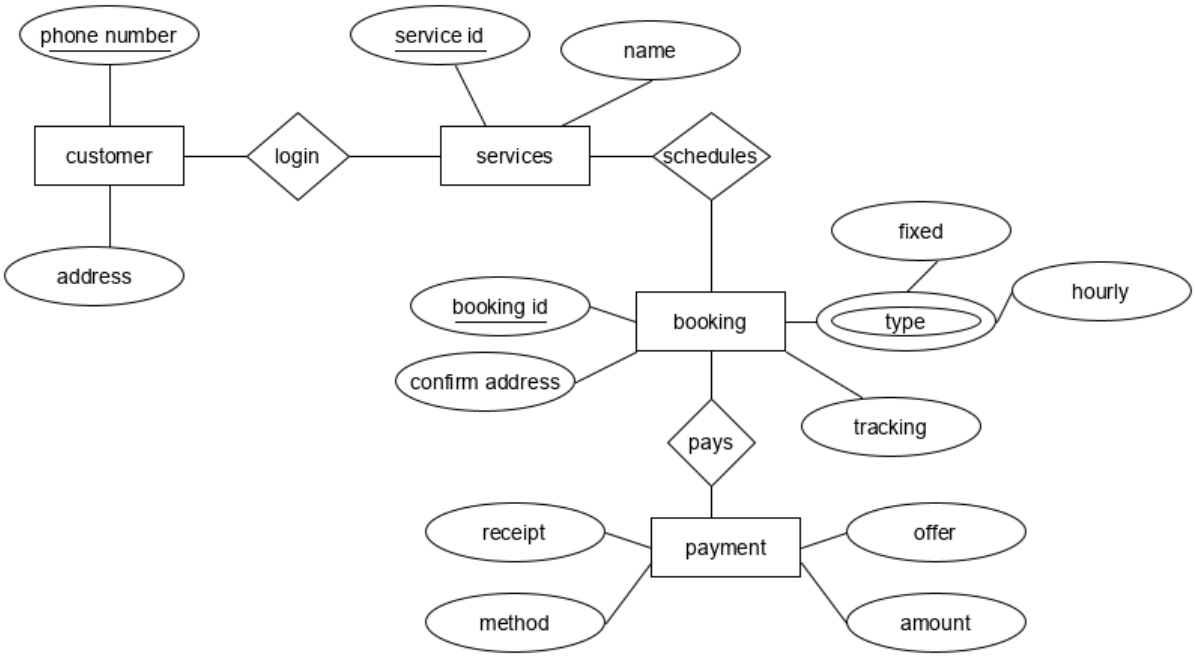
2.5 General Constraints

A full internet connection is required for OSS.

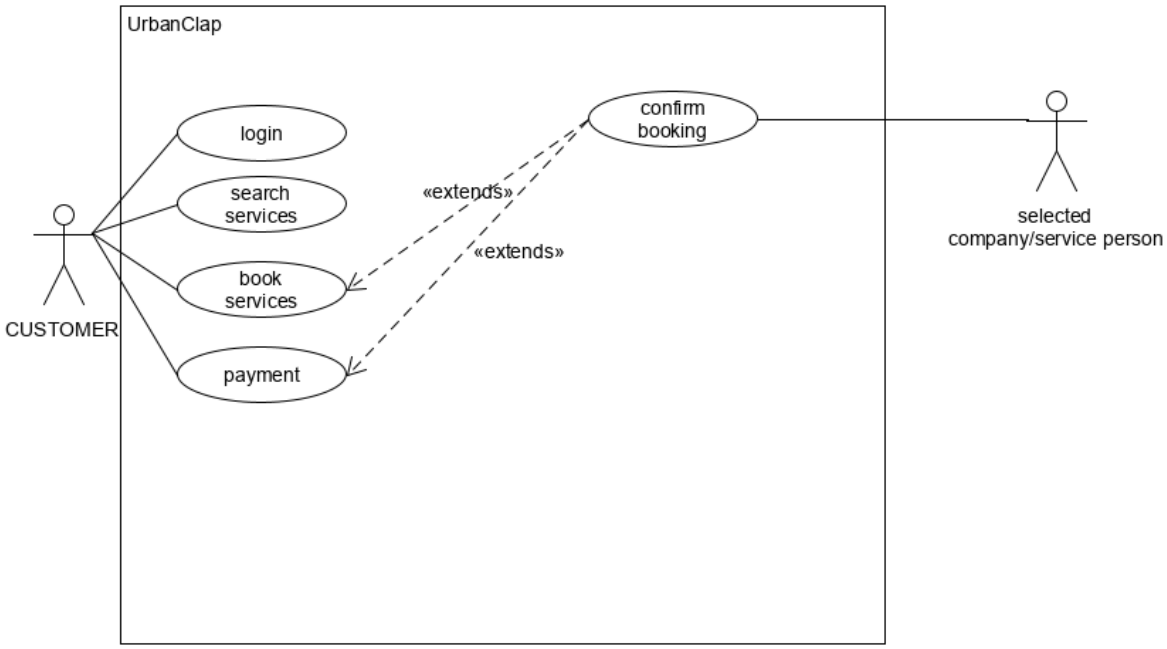
2.6 Assumptions and Dependencies:

Working of OSS need Internet Connection.

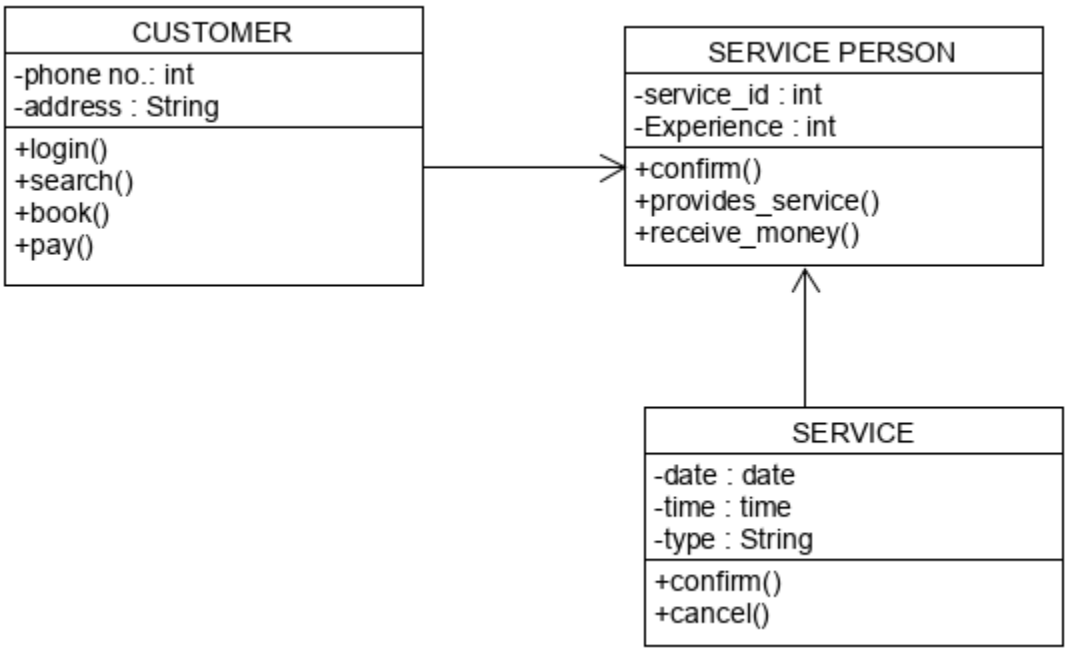
E-R diagram: 2



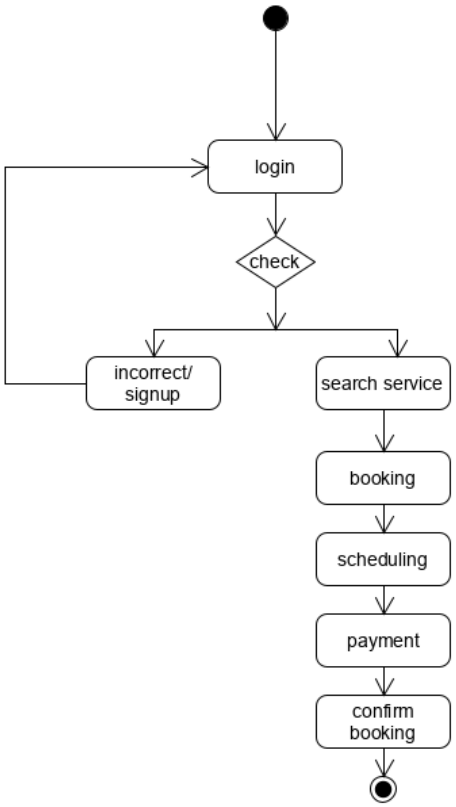
use case diagram: 2



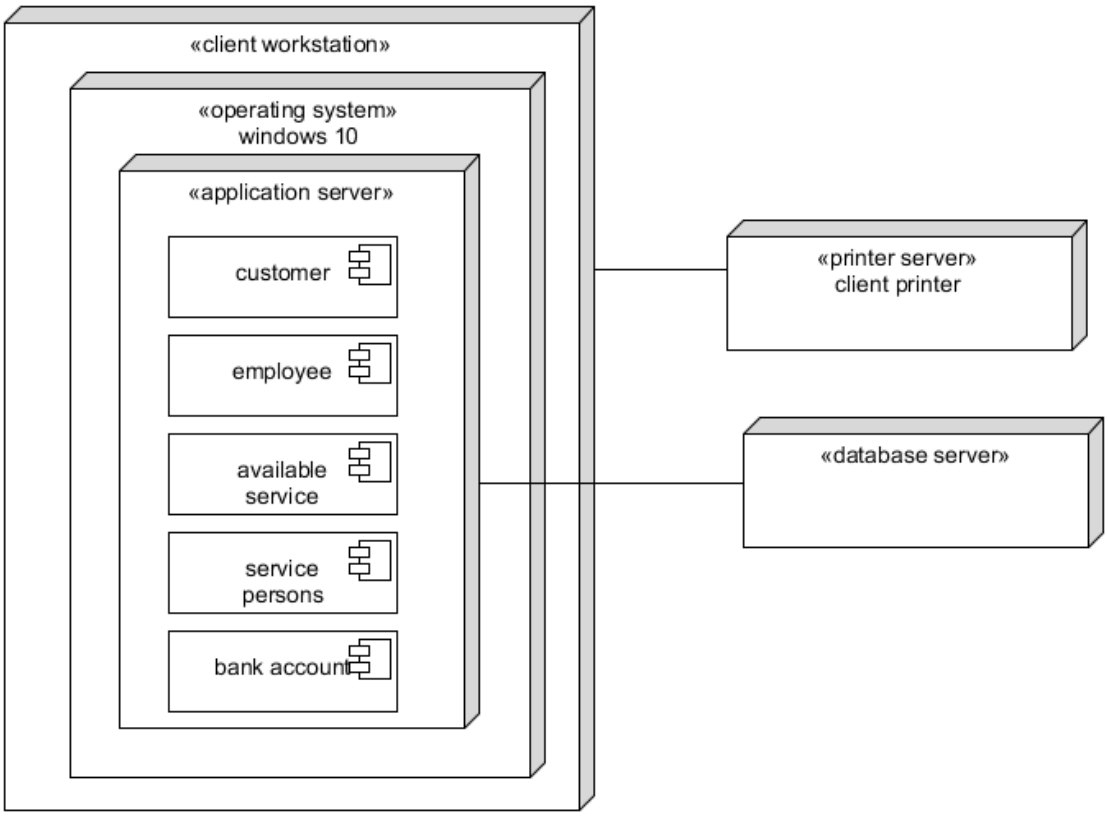
Class diagram: 2



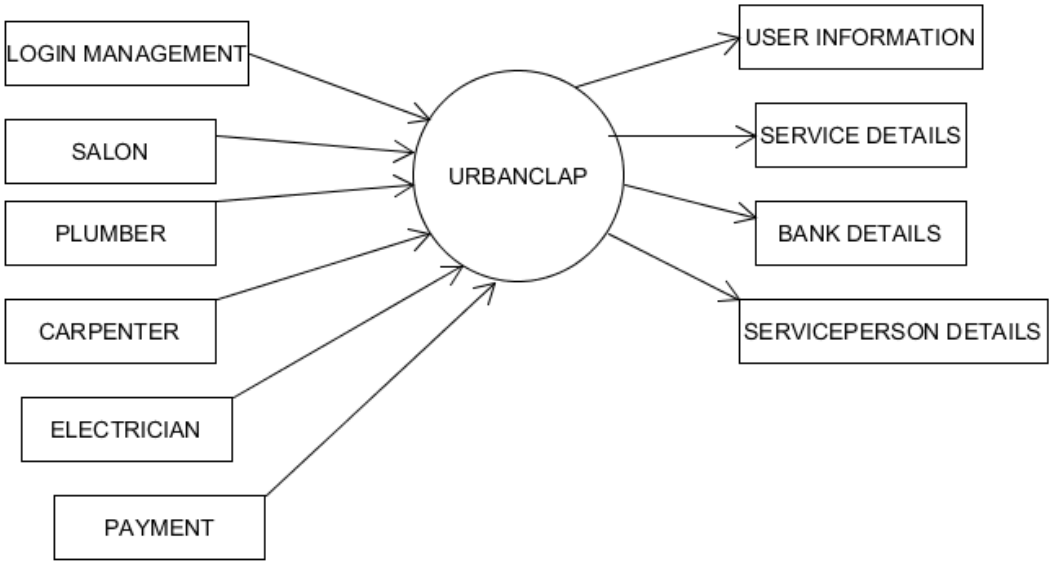
Activity diagram: 1



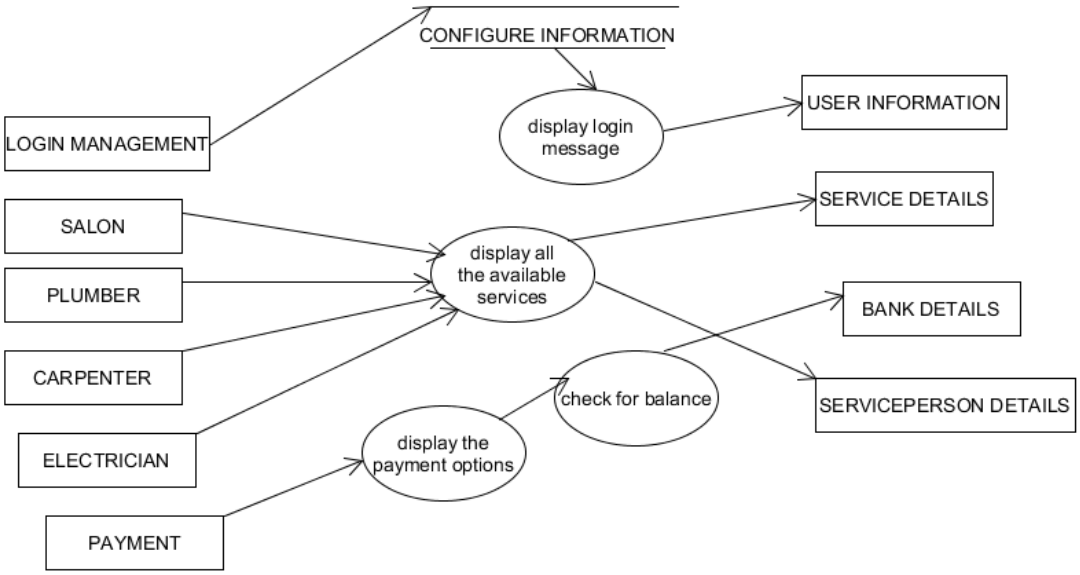
Deployment diagram: 1



Level 0 DFD: 1


LOGIN MANAGEMENT
SALON
PLUMBER
CARPENTER
ELECTRICIAN
PAYMENT
URBANCLAP
USER INFORMATION
SERVICE DETAILS
BANK DETAILS
SERVICEPERSON DETAILS

Level 1 DFD 1


CONFIGURE INFORMATION
LOGIN MANAGEMENT
display login message
USER INFORMATION
SALON
PLUMBER
CARPENTER
ELECTRICIAN
PAYMENT
display all the available services
SERVICE DETAILS
BANK DETAILS
SERVICEPERSON DETAILS
check for balance
display the payment options

3. Specific Requirements

This section contains the detailed requirements. In this section, the users of "UrbanClap" are referred to customer Users of other sections are referred Service Person.

3.1 External Interfaces

1) GUI (As described in table1)

The software provides a good graphical interface for the user and the administrator so that they can operate on the system, performing the required tasks such as Selecting, updating, paying and viewing the details of the booking details.

- It allows user to view quick reports like feedback, punctuality in between particular time.
- It provides verification and search consumer right services.
- The user interface must be customizable by the administrator.
- All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined.
- The design should be simple and all the different interfaces should follow a standard template.
- The user interface should be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module.

2) Login Interface:

In case the user is, not yet, registered, he can enter the details and register to create his account. Once his account is created, he can 'Login' which asks the user to type his Phone number. If the user entered either Phone number incorrectly then an error message appears.

3)Search:

The customer or service provider can enter the type of booking he is looking for and the work he is interested in, then he can search for the required work by entering the required service name.

4)Categories View:

Categories view shows the categories of available service provider, time slots and provides ability to the Admin to add/edit or delete category from the list.

5) Admin's Control Panel:

This control panel will allow Admin to add/remove users; add, edit, or remove a resource and manage lending options.

3.2 Functions

R.1: Register

- **Description:** First the user will have to register. There are two different type of users.
- **The Service Staff**: The manager has to provide details about the name of Service person, address, phone number, id.
- **Regular Person:** The user has to provide details about his/her name of address, phone number, payment details.

R.1.1: Sign up

- **Input:** Detail about the user as mentioned in the description.
- **Output:** Confirmation of registration status and a membership number and password will be generated and OTP to the user.
- **Processing:** All details will be checked and if any error is found then an error message is displayed else a membership number and password will be generated.

R.1.2: Login

- **Input**: Enter the membership number provided.
- **Output:** User will be able to use the features of software.

R.2: Manage Service by user.

R.2.1: Services.

- **Description:** List of services will be displayed along with data and time and available service person.

R.2.2: Search

- **Input:** Enter the name of service required.
- **Output:** List of services related to the keyword.

R.2.3: Services

- **State:** displays the services user wants to book.
- **Input:** book the required service that the user wants.
- **Output:** confirmation for booking and apology for failure in issue.
- **Processing:** if the selected booking is available then confirmed booking will be issued else error will be displayed.

R.2.4: Re-Booking

- **State**: Booking is issued and is about to reach the date.
- **Input:** Select the booking to be renewed.
- **Output:** confirmation message.
- **Processing:** If the issued booking is already reserved then the error message will be displayed otherwise the booking will be confirmed.

R.2.6: Reserve booking

- **Input**: Enter the details of the booking.
- **Output:** Booking successfully reserved.
- **Description:** If a booking is issued by someone then the user can reserve it, so that later the user can issue it.

R.3 Manage booking by Admin

R.3.1 Update details of bookings

R.3.1.1 Add bookings

- **Input**: Enter the details of the bookings such as: name, phone number, address and the time.
- **Output:** confirmation of the requested service.

R.3.1.2 Cancelation of Service

- **Input:** scheduled booking is cancelled.
- **Output:** Advance payment is kept as fine.

3.3 Performance Requirements

Response Time

As soon as the user books a service, he/she is contacted back within an hour.

Throughput

The number of transactions is directly dependent on the number of users, the users may be the Service person, employees of the company and also the people who use the Application for checking-out bookings, and checking online availability.

Capacity

The system is capable of handling more than 10,000 users at a time.

Resource Utilization

The resources are modified according the user requirements and also according to the services requested by the users.

3.4 Logical Database Requirements

The software should be able to keep a check on the user history like what kind of bookings he/she they prefer to use and by providing this particular data to the admin. It should be able to reflect those suggestions, the next time user login to the application. This will surely be helpful to the user as he/she can easily choose their preferred services.

3.5 Design Constraints

Software Language Used

The languages that shall be used for coding the Urban Clap are Active Server Pages (ASP), Java Servlets, Java Server Pages (JSP), HTML, JavaScript, and VBScript. For working on the coding phase of the Urban Clap, the Internet Information Services (IIS) Server needs to be installed.

Development Tools

Will make use of the available Java Development Tool kits for working with Java Beans and Java Server Pages. Also, will make use of the online references available for developing programs in ASP, HTML and the two scripting languages, JavaScript and VBScript.

Class Libraries

Will make use of the existing Java libraries available for JSP and Servlets. Also, we need to develop some new libraries for the web-based application. Also, will develop new programs using ASP and scripting languages.

3.5.1 Standards Compliance

All the user's entries will be stored in the admin's database so the admin can easily keep a check on the all the entries of different user's, hence this will make our software more efficient and compatible. A specific money is also needed to achieve such effectiveness in the software. Small meetings will also be scheduled between the developer team and the client.

3.6 Software System Attributes

System attribute requirements such as those for performance, security, modifiability, reliability, and usability have a significant influence on the software architecture of a system. Architects need to understand their designs in terms of quality attributes. For example, they need to understand whether they will achieve deadlines in real time systems, what kind of modifications are supported by their design and how the system will respond in the event of a failure. There are large and thriving attribute communities that study various quality attributes but they each have their own language and sets of concepts.

3.6.1 Reliability

To apply engineering knowledge and specialist techniques to prevent or to reduce the likelihood or frequency of failures of the system.

There will be a team to identify and correct the causes of failures that do occur despite the efforts to prevent them and to determine ways of coping with failures that do occur, if their causes have not been corrected, to apply methods for estimating the likely reliability of new designs, and for analyzing reliability data.

The reason for the priority emphasis is that it is by far the most effective way of working, in terms of minimizing costs and generating reliable products.

The primary skills that are required, therefore, are the ability to understand and anticipate the possible causes of failures, and knowledge of how to prevent them. It is also necessary to have knowledge of the methods that can be used for analyzing designs and data.

3.6.2 Availability

System or the website is available at online platforms and is also available as an android and iOS (As described in table1) application.

3.6.3 Security

The system is a public website; therefore, all the content will be available for the users.

3.6.4 Maintainability

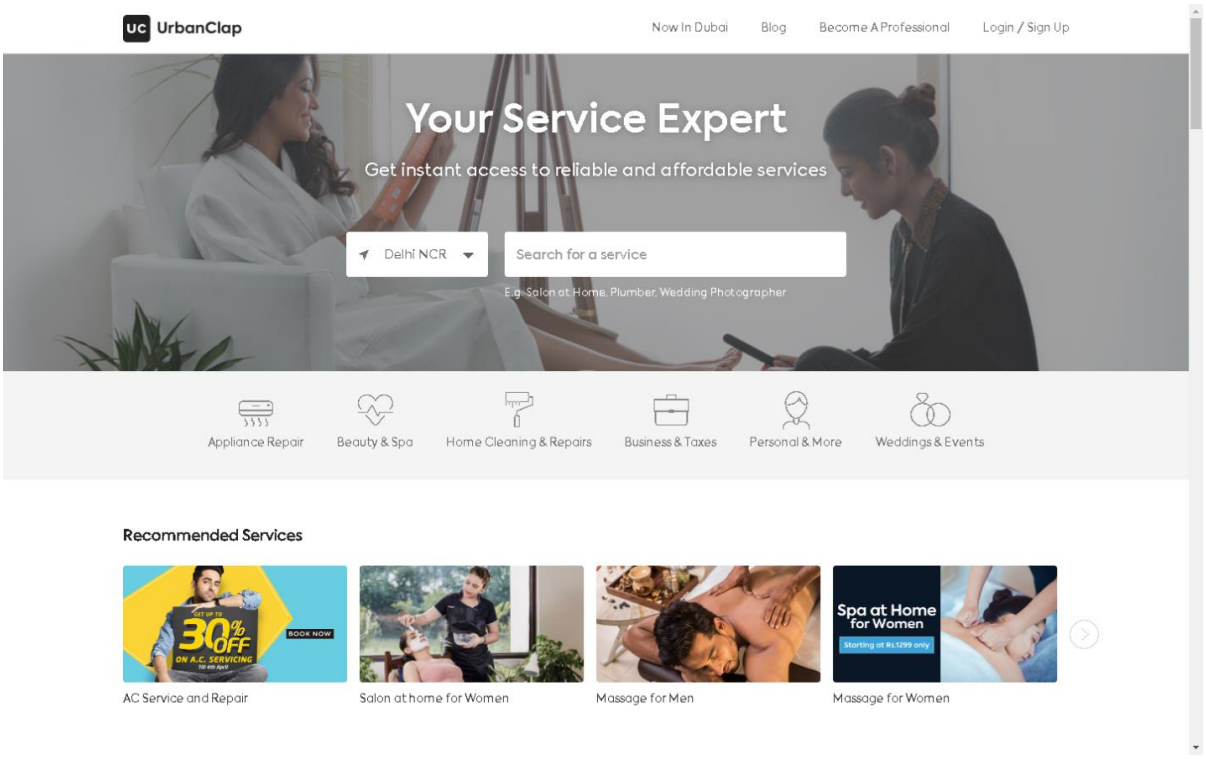
There will be no maintained requirement for the software. The database is provided by the end user and therefore is maintained by this user.

3.6.5 Portability

The software we will be developing will be compatible with numerous platforms. We will also make our source code translatable to other programming language in order to increase the portability of the software. The client will be able switch the software to different number of systems without failure.

4. Change Management Process

When the above-mentioned system will be put in use, the end users will demand for some updation or they may require some changes as well so for that purpose there will be an entirely team working for them, they can simply approach that particular team either by dropping an e mail or a phone call as the number will be provided. After this our whole team will discuss regarding that particular issue whether that demand can be fulfilled or nor and if can then what are the possible ways to implement that particular updation.



Homepage 1

5 Testing Phase

5.1. Introduction

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest

Testing is the process of exercising software with the intent of finding (and ultimately correcting) errors. In fact, because Web-based systems and applications reside on a network and interoperate with many different operating systems, browsers (residing on a variety of devices), hardware platforms, communications protocols, and "backroom" applications, the search for errors represents a significant challenge

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Testing is a set of activities that can be planned in advance and conducted systematically.

For this reason a template for software testing—a set of steps into which you can place specific test case design techniques and testing methods—should be defined for the software process. A number of software testing strategies have been proposed in the literature.

All provide you with a template for testing and all have the following generic

Characteristics:

- To perform effective testing, you should conduct effective technical reviews. By doing this, many errors will be eliminated before testing commences.
- Testing begins at the component level and works "outward" toward the integration of the entire computer-based system.
- Different testing techniques are appropriate for different software engineering approaches and at different points in time.
- Testing is conducted by the developer of the software and (for large projects)an independent test group.
- Testing and debugging are different activities, but debugging must be accommodated in any testing strategy.

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. A strategy should provide guidance for the practitioner and a set of milestones for the

manager. Because the steps of the test strategy occur at a time when deadline pressure begins to rise, progress must be measurable and problems should surface as early as possible.

Software testing is one element of a broader topic that is often referred to as verification and validation (V&V). Verification refers to the set of tasks that ensure that software correctly implements a specific function. Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. Boehm states this way:

Verification: "Are we building the product right?"

Validation: "Are we building the right product?"

5.2. Types of testing

System Testing

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

Interface Testing

Interface Testing is defined as a software testing type which verifies whether the communication between two different software systems is done correctly. A connection that integrates two components is called interface. This interface in a computer world could be anything like API's, web services, etc. Testing of these connecting services or interface is referred to as Interface Testing.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. Test case design techniques that focus on inputs and outputs are more prevalent during integration, although techniques that exercise specific program paths may be used to ensure coverage of major control paths.

Validation Testing

Validation testing provides final assurance that software meets all informational, functional, behavioral, and performance requirements. The last high-order testing step falls outside the boundary of software engineering and

into the broader context of computer system engineering. Software, once validated, must be combined with other system elements (e.g., hardware, people, databases).

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components.

Smoke Testing

Smoke Testing is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program are working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application so that the QA team does not waste time installing and testing the software application. In Smoke Testing, the test cases chose to cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.

Regression Testing

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the new code changes are done.

Acceptance Testing

Acceptance is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment. The main purpose of this testing is to validate the end to end business flow. It does NOT focus on cosmetic errors, Spelling mistakes or System testing. This testing is carried out in a separate testing environment with production like data setup. It is a kind of black box testing where two or more end users will be involved.

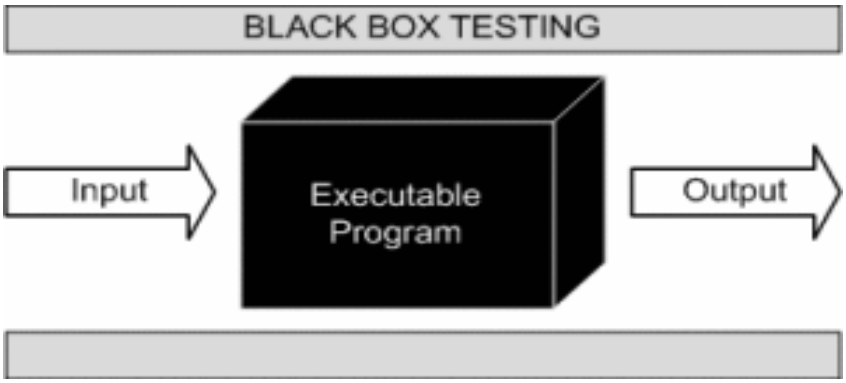
Sanity Testing

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing. The objective is "not" to verify thoroughly the new functionality but to determine that the developer has applied some rationality (sanity) while producing the software.

Black Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Test design techniques include Equivalence partitioning, Boundary Value Analysis, Cause-Effect Graphing.

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



Black Box Testing (Diagram)

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

Incorrect or missing functions

Interface errors

Errors in data structures or external database access

Behavior or performance errors

Initialization and termination errors

Black box test design technique: Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

Black Box Testing method is applicable to the following levels of software testing:

System Testing

Integration Testing

Acceptance Testing

The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use

White Box Testing

White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Test design techniques include Control flow testing, Data flow testing, Branch testing, Path testing.

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

White-box testing is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases. Using white-box testing methods, you can derive test cases that

(1) Guarantee that all independent paths within a module have been exercised at least once,

(2) exercise all logical decisions on their true and false sides,

(3) Execute all loops at their boundaries and within their operational bounds, and (4) exercise internal data structures to ensure their validity.

Basis path testing is a white-box testing technique first proposed by Tom McCabe. The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

Working process of white box testing:

Input: Requirements, Functional specifications, design documents, source code.

Processing: Performing risk analysis for guiding through the entire process.

Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.

Output: Preparing final report of the entire testing process.

White-box testing's basic procedures require the tester to have an in-depth knowledge of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created.

5.3 TEST CASES

- During the payment process, try to change the payment gateway language.
- Check what happens if payment gateway stops responding during payment.
- Check what happens if payment gateway stops responding during payment.
- During the payment process check what happens in the backend.
- Check the Database entries whether they store credit card details or not.
- Check settings of pop-up blocker, and see what happens if a pop-up blocker is on and off.
- Check on successful payment, a success code is sent to the application and a confirmation page is shown to the user.
- After successful transaction check if the payment gateway returns to your application.
- Unless you don't have an authorization receipt from the payment gateway, service should not be booked.
- Check the amount format with currency format.
- Check if each listed payment option opens the respective payment option according to specification.
- Verify the default option for debit card shows card selection drop down menu
- After successful payment, test all the necessary components, whether it is retrieved or not.
- During payment process check what happens when session ends.
- Check what happens if payment process fails.
- During the payment process check security pages and error pages.
- Between payment gateway and application check buffer pages.
- Verify whether the transaction processes immediately or processing in hand to your bank.
- Check all format and messages when successful payment process,
- Inform the owner for any transaction processed through e-mail. Encrypt the content of the mail.

6. Document Approvals

Approved by: Ms. Sirat Kaur

Date:22nd of April,2019

Signature: ……