COMS3008:
Software Design

FLCKS Restaurant Booking System
Architecture Document

Team Members

Mthulisi Leslie Zimba:	**570937**
Kgopotso Dilebo:	**715636**
Lethabo Nkabinde:	**722211**
Christopher Mashele:	**732475**
Fortune Ndlovu:	**731603**

WITS
UNIVERSITY

School of Computer Science and Applied Mathematics
University of the Witwatersrand
South Africa

10 October 2016

Contents

1 Introduction

1.1 Scope

The following document pertains to, or describes the architecture of a mobile restaurant booking system application created by FLCKZ, which works as a booking system for restaurants . The application gives you access to restaurant menus, allows one to book tables and order meals,also check table availability and also rate/share about the service received from the restaurant.
Certain topics will be discussed at different levels of the Architecture, being relevant to different parties, details to which parties are applicable will be clearly stated.

The system works on android devices and is downloadable via the Google Play Store on your mobile device.

1.2 Acronyms and Abbreviations

Some Acronyms and Abbreviations used in this document are mentioned below:

- FLCKZ: Design team name derived from the names of the group Members (Fortune, Lethabo,Chris,Kgopotsho, Mthulisi (Zimba))
- GUI: Graphic User Interface
- UML: Unifed Modeling Language

1.3 Definitions

Some Definitions explained:

- PHP: A server-side scripting language designed for web development.
- Event-Driven Architecture: An Architecture development design Technique
- Event-mediator: The event mediator receives the initial event and orchestrates that event by sending additional asynchronous events to event channels to execute each step of the process.
- Event-process: which listen on the event channels, receive the event from the event mediator and execute specific business logic to process the event.

1.4 References

2 Architectural Representation

This document details the Architecture using the views defined in the "4+1" model, but using the RUP naming convention. The FLCKS application will be using the following views:

- **Logical View**

  Audience: Designers
  Area: Functional Requirements: discribes the design object model. Also describes the most important use-case realizations.
  Related Artifact: Design Model

- **Process View**

  Audience: Integrators
  Area: Non-Functional Requirements: discribes the concurrency and synchronization aspects.
  Related Artifact: None

- **Implementation View:**

  Audience: Programmers
  Area: Software Components: Describes the layers and subsystems of the application
  Related Artifact: Implementation Model.

- **Deployment View:**

  Audience: Deployment Managers
  Area: Topology: discribes the mapping of the software onto hardware and shows system distributed aspects.
  Related Artifact: Deployment Model.

- **Use-Case View:**

  Audience: Stakeholders
  Area: Scenerios and use cases
  Related Artifact: Use-case Model and Use-case documents.

3 Architectural goals and constraints

This document aims to show how our application will work and to form a blueprint which will enable one to navigate and explore easily through the Application to give one a better idea as to what our Application does.
This section describes the software requirements and objectives that have a significant impact on the architecture.

3.1 Software Decisions

The System that we are creating is intended to perform well in an ever-changing environment because it is intended on being used on a daily basis, keeping in mind that it also needs to be very secure and thus we wouldn't want any data to be leaked/published, because confidential information will be dealt with. We also want a high performing application. We expect the application to be accessed by a large number of people and thus would like to be able to upgrade/ scale up if needs be in order to accommodate large volumes of users in the future.

With that in mind we have decided to use the event-driven architectural style because it performs well in the key factors required for the application. We will be using the Mediator Topology in the creation of the Application due to a number of different factors, which will be explained throughout the document.

Our Application will have two main users, namely the Client and the Administrator, thus we found it fitting that since the Administrator will be at work most of the time, that they should access their Administrative privileges using their office computer. Thus no cost will be put to the company in terms of mapping the software to hardware, they just use their current Office PC.
The Client on the other Hand will only be able to access the system via an android device. This decision was made due to recent statistics that show the ever increasing number of android devices available and the number of people who use android devices today.

Depicted is the Architecture Decision diagram, where the black blocks represent the type of functionality and the box below depicts the domain to which the functionality will be accessed.

3.2 Technical Platform

The FLCKZ App will be deployed onto a Firebase server since it was the best platform available to us.

3.3 Transaction

The FLCKZ App is transactional, leveraging the technical platform capabilities. Transaction management model of the lamp will be reused intensively.

3.4 Security

The system must be secured, so that a customer can view events on the App. The application must implement basic security behaviours:

- Authentication: Login using at least a user name and a password

- Authorization: according to their profile, on-line customer must be granted or not to perform some specific actions, they should have different granularities of access to the App depending on their profile.
  For internet access, the following requirements are mandatory
- Confidentiality: sensitive data must be encrypted (passwords)
- Data integrity: Data sent across the network cannot be modified by a tier
- Auditing: Every sensitive action can be logged
- Non-repudiation: gives evidence a specific action occurred

3.5 Persistence

Data persistence will be addressed using a relational database.

3.6 Performance

The login process must take no longer than 5 seconds.

3.7 Reliability and Availability

The availability of the system is a key requirement by nature, as it is a selling system. The candidate architecture must ensure failover capabilities. Targeted availability is 12/5: 12 hours a day, 5 days a week The time left (8 hours) is reserved for any maintenance activities.

4 Scenarios

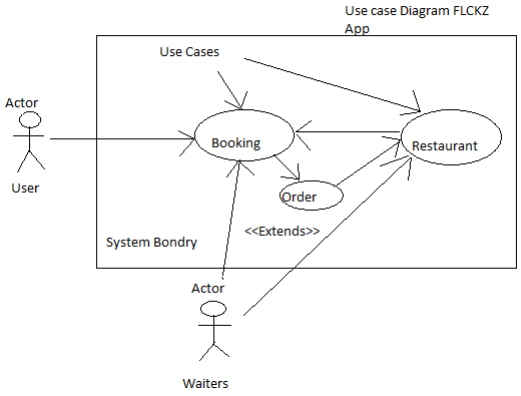
4.1 Types of Scenarios

True there are many different scenarios that could take place, as mentioned in the topic discussed above, and not all can be mentioned in this documentation. But a brief overview of the most basic and common scenarios that will be considered, due to the nature of the system as a whole. Here is a list of the top scenarios that will be fulfilled by the System:

- Register: This scenario is the first experience you have with the Application, it enables you to create your user profile and adding your details to the local host database system.
- Login: Once you have a profile, this scenario allows you to access the Application and gives a client the ability to start performing the different tasks given to them by the Application.
- Logout: Once you have completed your given tasks, then it allows you to exit your profile and login another profile.
- View menu: This scenario will be one of the most common because if you dont want to book a table, you can still use this scenario in order to keep up to date with the current meals and specials offered by the restaurant.
- Book table: This is also a common scenario and was mentioned in detail in the above section.
- Order meal: This can be done by selecting the meals you fancy in the menu and clicking requesting it by clicking on the order button.

4.2 Booking a Table and Order Meals

A customer accesses the FLCKZ App and requests to book a table in a given restaurant. The customer then specifies the occasion and the amount of people need to be accommodated on that table.Then, the customer confirms his booking. After that, if the table is available then a notification is sent back to the customer. And thus the customer will be given the option to order a meal.
This is depicted in the following use case diagram



These are some of the functionalities we are working on:

- Rate service: This will enable you to rate the service you recieved both by the restaurant and the Application in store. More details will revealed as the progress of the Application unfolds.
- Share experience on social media: Brag to your friends on your experience with the Application and restaurant.

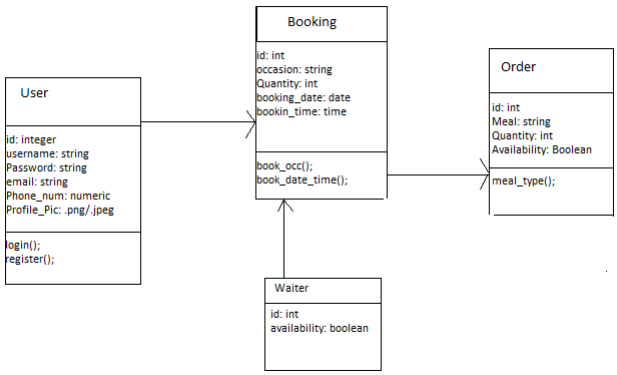
4.3 Use-Case Realization

Refers to how design elements provide the functionalities identified in the significant use-cases.

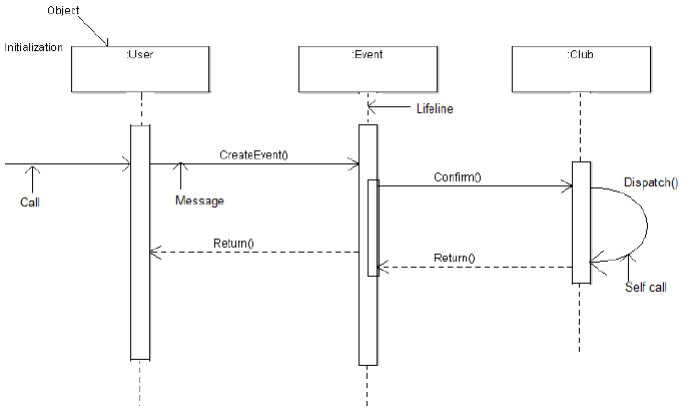
5 Logical View

5.1 Overview

The FLCKZ application is divided into layers based on the N-tier architecture. The layering model of the FLCKZ application is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.



5.2 Architecture Layer Dependency



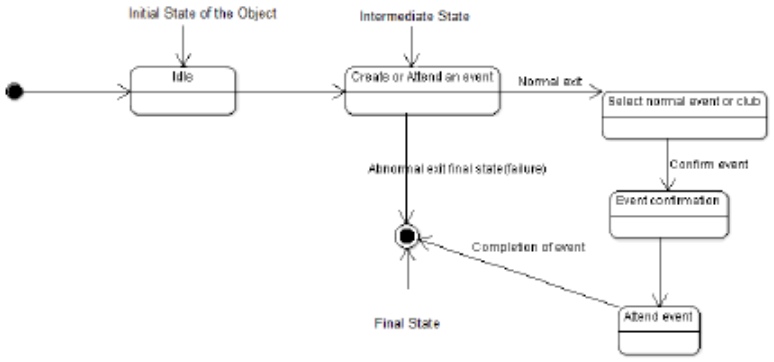
Each layer has specific responsibilities.

- The presentation layer deals with the presentation logic and the pages rendering
- The control layer manages the access to the domain layer

- The resource layer (integration layer) is responsible for the access to the enterprise information system (databases or other sources of information)
- The domain layer is related to the business logic and manages the accesses to the resource layer.
- The Common Elements layer gathers the common objects reused through all the layers

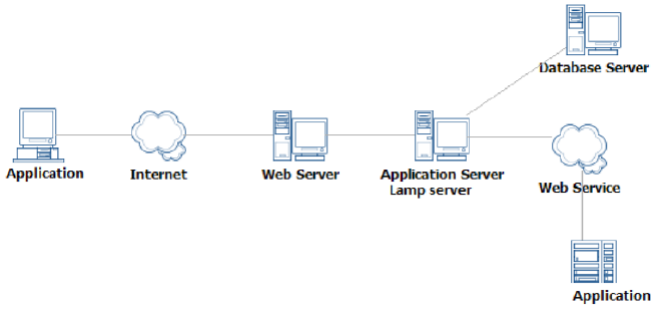
6 Process View

The following diagram describes the state of the program when the user decides to book a table. The system will be in idle until the an occasion is selected. Then if the action is unsuccessful then the program will move to the final state. Otherwise the program will exit normally to the select the quantity of people - when and what time the table is booked for. Then on confirmation, the state will change to order confirmation. Then on to confirm order. After all that is done, the program will move to the final state.



7 Physical View

The application is Android based. It is assumed to register it's users using a database stored on the lamp server. So the application would need to connect to the internet first, then be directed to web servers that are going to grant it access to the lamp server. Then from the lamp server the application will access the database. Redirect everything that it got from those databases back to the web services then back to the application.



8 Deployment View

8.1 Layers

8.1.1 Presentation Layer

The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface

8.1.2 Control Layer

The Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate.

8.1.3 Resource Layer

The Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems (Database, external services, ERP, etc. . .)

8.1.4 Domain layer

The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model.

8.1.5 Common Elements Layer

The Common Element layer contains the components re-used within several layers.

The Development view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files). Usually, the layers of the Development view do fit the layering defined in the Logical view It is unnecessary to document the Development view in great details in this document. For further information, refer to the On-line Catering Service 1.0 workspace in Rational Software Architect.