Project M-music
A personalised web-site accessible through multi-clients devices using Java and XML technology

Please go to page 121 for code exmaple

Vishal Sakaria 9902757

Computing with Marketing BScH

Abstract

The project is aimed at exploring new technologies and business matters that are influencing the wireless world. These technologies are then used to build a web site that can be accessed by wireless devices, as well as a desktop PC. When building a web application for a wireless device memory, display size and the amount of content sent must be taken into consideration; this project shows how these issues can be tackled with technologies such as XML.

UML methodology is used to design the system, with an additional web extension to help create a clearer picture of new web technologies.

Contents

Acknowledgments

During my last year at London Guildhall University I found new sides to me which allowed me to develop as a human being, once I discovered these sides I soon realised were they had come from, the following is a thank you to those people who have help me become me.

Firstly I must thank my father, Denish Sakaria, although I have no close relationship with you, I must thank you for bringing me onto this earth and more importantly I thank you for all the things I have inherited from you, your genes have allows me to see the world in a more understanding and dynamic way. All due respect must be give to the Kotecha family who gave me the stability I needed as a young child. A special thank you to Sunil Kotecha, it was you who took me under your wing when I had no direction and it is you who has influenced me in a way that will give me happiness, your always there for me, a gift I will never forget. Recently I have come to understand the importance of emotions and also where my emotional fuel comes from, my next acknowledgment therefore, and one, which is closet to my heart, goes to my mother. A beautiful woman who through thick and thin has always loved me and wanted the best for me, due to your strong emotional genes I have the fuel to do anything in my life.

Lastly I must thank Nirmal Kumar for giving me such sound advice and support during this project, you help me build the bridge to the wireless world.

Section 1 – Introduction

Overview:

The objective of this study is to produce a report consisting of various design stages. In order to deliver a valuable application to the client, a number of steps will be followed, considerable research demonstrated and a methodological approach in a computerised application which will be demonstrated and tested. The stages of the report will be planned and executed in a controlled and tightly managed environment, with expert opinion being sought where necessary and investigation at all times being thorough. Stages will be divided into subheadings and diagrams and models displayed where appropriate.

Background

Having completed four successful units in Marketing, and have the view to work in such an industry on completion of my degree. I have chosen to base my project on future factors affecting both marketing and the computing world. Wireless internet is likely to be the next step of the information revolution, and as over 46 million people in the UK alone have a wireless device, the prospects for the Third Generation (3G) of mobile communication is an exciting and profitable stance. Personalisation is a marketing concept which can now be executed with greater reality, Amazon.com for example use Personalised portal, displaying information personalised for the user, it assumes that the final result will deliver the appropriate message to the appropriate person at the appropriate time. This is an effective method of commerce for wired desktop PC, a PC however normally is not personal to just one individual, many people in an office or a family use the same computer. An Internet PC is not always available; you can't take the Internet connection with you when you pick up your children from school or when you go to visit your friend. In the mobile world however each individual owns their own mobile phone, hence making it their own personal device and because it is wireless it can be taken almost anywhere. The prospects of personalisation is evident through the personal digital assistant and mobile phones, even today with the vast restriction of mobile phones people still aim to personalise them, they choice there own ring tones, icons, setting and even their phone book is personal to them alone, this just emphasises the prospect of 3G devices in conjunction with personalised portals

Business consideration with 3rd Generation technology.

Europe has over 185 million Internet users; of that 12.5 million exist in the UK. These figures compared with the number of mobile users, (46 million within the UK), shows the potential of the mobile revolution. It is therefore important to gain some programming and marketing knowledge about factors affecting this revolution. E-commerce has been the first step in creating revenue over the internet, many lessons have been learnt which will be used in m-commerce which also reaps some benefits. For example, impulsive buying which is not possible with e-commerce may take effect in the mobile industry. But there are other great advantages which make the mobile industry important. Companies will be able to give value added service, a marketing theory implemented in highly competitive markets, which is true for most e-commerce companies. A company will be able to differentiate itself from competitor by targeting market groups more efficiently and effectively, advertising a women's product for example will only be sent to devices meeting the correct profile, this would save money and the customer would be more likely to acknowledge the advertisement. Other valued added service could include flight cancellation details which could be sent to customers hours before their flight was due to leave, then the option would be given to reschedule, once you reach your destination a mobile device will be able to give you information on the restaurants and entertainment in the area, also in the event of an accident the rescue service will know your location even if you don't, the possibilities are endless.

Another trend mobile technology which hold great prospects is remote transaction environment, transactions are done over a digital mobile network and the participants is not, in most cases so relevant. Most transaction is conducted with menu driven applications enabling for example more payments options. Remote transactions range from online purchases and banking to more impulsive buying like downloading digital content as mention before. In addition to remote transaction, local (or proximity) transaction environment are available, local transaction are usually initiated over short range wireless technology surrounding, such technologies such as Bluetooth developed by Erickson would be integrated in the mobile device and the electronic vendor terminal. The performance requirements, such as speed and easiness would be high, allowing simpler cash free transactions and would promote impulsive buying. Products such as drinks, fast food, travel tickets and most other fast moving consumer goods would all benefit from such technology.

Privacy consideration with the internet and 3rd Generation

When you're on the Internet, all the steps you take, however small or trivial, are recorded and stored somewhere, you just don't know about it. An example of how the Internet plays fast and loose with your privacy, is the selling of on-line customer profiles, as discussed earlier a profile of personalised information can be stored by a company in what is know as session tracking, session tracking can record every click you make whilst on a web site. Although this is beneficial as it allows companies to build knowledge on customers to promote goods and services that you might find interesting. Companies can also sell your profile to other companies that might want to promote their goods directly to you. This is considered an invasion of privacy and many people are finding this sacrifice of personal privacy as a tradeoffs they're being forced to make when using the Internet. One way of getting around this is to disenable Cookies; Cookies are very small text files placed on your hard drive by a Web Page server. It is essentially your identification card, and cannot be executed as code or deliver viruses. It is uniquely yours and can only be read by the server that gave it to you. Cookies can store information about you, for example username and password allowing automatic login, however this reduces privacy and although disenabling them may reduce richer sessions many people still do so.

Section 2 - Objectives

It is not the intension of this project to design a fully working commerce web-site with intensive graphical user interface but rather to build a prototype demonstrating the ideas and techniques behind wireless communication and personalisation. The web site will be built so multi clients can access content, that is WAP (wireless application protocol) enable phone, a Java enabled device and a computer with a web browser (Internet Explorer 5), the site will be multi-tiered; that is it will use different levels of implementation. The bottom level will be the data level and will involve a database; the middle tier will deal with business logic by performing actions or functions on the data extracted from the database. The final top level, the client tier will deal with the clients user interface and will be implemented in accordance to the device type browser specification, see figure 2 for details.

The next objective of the project is to implement some type of personalisation method, which will enrich the user's session; the aim is to allow users to login freely and securely and view information personal to them. Users will be able to create a profile of their interests and thus view information derived from their profile, user should also be able to change their profile as and when they wish, it is also the responsibility of the web site to give users information which relates to their profile and thus expanding the session information given to the users.

As mentioned it is not the main objective of this project to produce a fully automated web site, however application content should be defined, with my interest in music and the idea of listening to any choice of music, anywhere, any time whilst on the move, brings me to choice a music web site. The site will be called m-music or Mobile music and will hold data on customer music interests.

Thus the overall objectives of the web site will be to;

- Allow multi clients types to access the web site.
- Allow users to create a profile.
- Allow user to view information in accordance to their profiles.
- Allow user to view information outside of their profile, but still related to their profile choice.

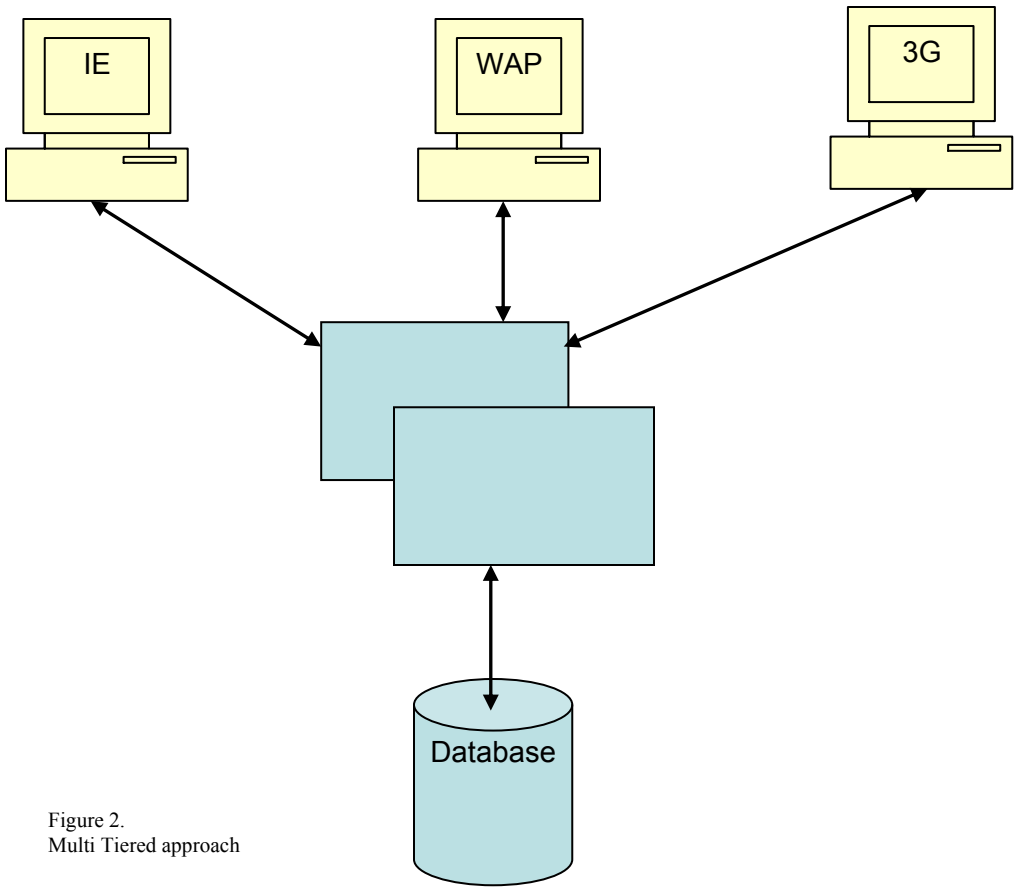


Figure 2.
Multi Tiered approach

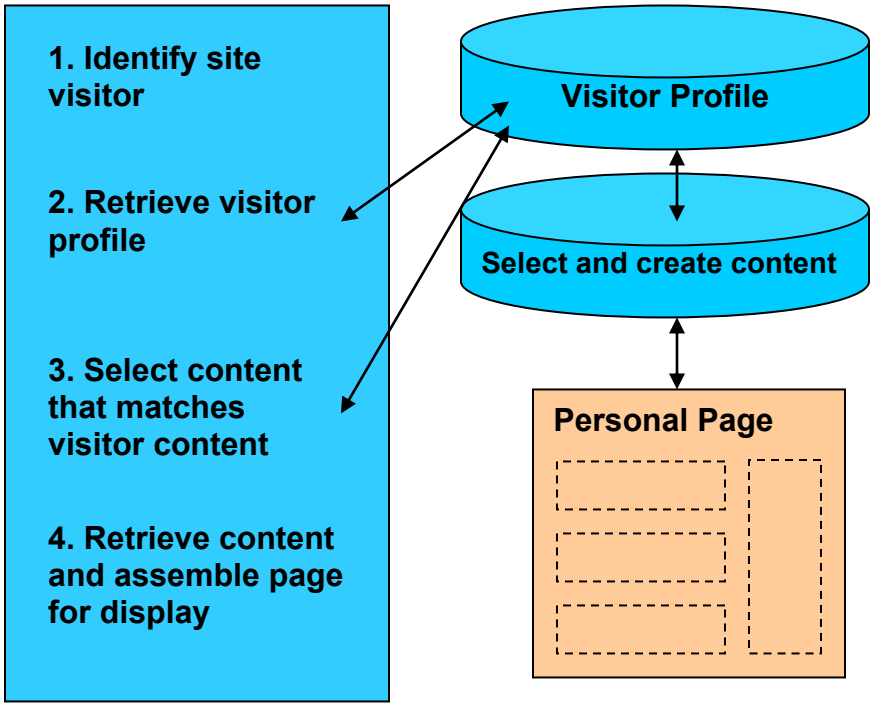


Figure 3, Step to personalisation

Section 3 – Technological and Technical options

This section introduces some technological issues to consider, it also outlines some techniques documented in papers written by larger organisations. It is not the aim of this section to go deeply into these options but rather to give an overview and reason for choosing them from a business perceptive, which would seem appropriate as the true nature of the technologies is unknown at this point. A detailed explanation of the technologies will be explained in following sections.

Technical issues of personalisation.

There is no doubt that a personalised web site requires some sort of dynamic content, although the index page may be uniform, once a user has logged in pre-coded static pages will be just to tedious and long winded to implement. It is for this reason server sided programming has become a standard for web design, technologies such as Microsoft's Active Server pages (ASP), Common Gateway Interface (CGI) and Sun's Java Server Pages (JSP) among others have been introduced. Such technologies allow web site to act in a multi tier fashion, users can input data at the highest level, client level, this data can then be passed on to a middle tier or the business level which can perform operation on the data and communicate with a database that exists at the lowest level, the data level. It is this interface between the top and bottom level where server sided programming exists. The middle tier or business logic and has access to all server resources such as emailing, other server programs and of course databases. Because server sided programs are actually implemented in a programming language, not just a mark-up language, dynamic content can be produced and sent to the user as different data is retrieved and sent to the program. It is therefore mandatory to use server sided programming to build dynamic personalised web sites.

Personalisation techniques

Personalization is a process of gathering and storing information about site visitors, analyzing the information, and based on the analysis, delivering the right information to each visitor at the right time, this approach can be seen in figure 3. Two major steps can be derived from this approach firstly collecting information and secondly analysing the data and giving a recommendation.

Collection visitor information

The objective of collecting visitor information is to develop a profile that describes visitor's interests or some other description of importance to the site owner. The most common techniques for collecting data are Explicit profiling, Implicit profiling and legacy data accesses.

Explicit profiling asks each visitor to fill out information or questionnaires. This method has the advantage of letting customers tell the site directly what they want to see.

Implicit profiling tracks the visitor's behavior. This technique is generally transparent to the visitor and is also called session tracking. Browsing and buying patterns are the behaviors most often assessed. The browsing pattern is usually tracked by saving specific visitor identification and behavior information in a cookie that is kept on the client side and updated at each visit. The buying pattern is available in the customer database situated on the server.

Legacy data accesses data from valuable profile information, such as credit applications and previous purchases. For existing customers and known visitors, legacy data often provides the richest source of profile information but requires high capital investment and analysis method to exploit the data.

Analysing Data

When the profile is available, the next step is to analyze the profile information in order to present the web page to the visitor. Making such recommendations is the most challenging step. Many techniques for presenting content and making recommendations are in use or under development. Rule-based and filtering techniques are the best known.

Rule based techniques provide a visual editing environment for the business administrator to specify business rules to drive personalisation. This requires the administrator, most likely with the help of a consultant, to figure out the appropriate rules. The rule-based approach provides a flexible mechanism to specify rules for business applications or marketing campaigns. For example ruled based mechanism could specify that when product X is assigned to the user profile product Y would be suggested to the user; for example if a user selects Jazz as a music preference Soul music would be suggested.

Filter mechanism use algorithms to analyses data and make recommendations, the three most common filter techniques today are simple filtering, content-based filtering, and collaborative filtering.

Simple filtering relies on a common group of users accessing the site, each group is assigned content and the page is displayed, an example of this in a business context is the assigning of different privileges to different departments of a business. From a commerce perspective different age groups or genders could be given different content.

Content-based filtering works by analyzing the content of the objects to form a representation of the visitor's interests. Generally, the analysis needs to identify a set of key attributes for each object and then fill in the attribute values. One example is a document filtering system that analyzes documents based on keywords. Recommending video movie purchases is another example of content-based filtering. Content-based filtering is most suitable when the objects are easily analyzed by computer and the visitor's decision about object suitability is not subjective.

Collaborative filtering collects visitors' interests either implicitly or explicitly and forms like minded peer groups, this information is used to learn more about each group, suggestions are then made not on the user previous choice or similar factors based on a business rule but on what similar people from similar group have decided.

All of these techniques if well implemented can produce powerful personalised pages however it is the content and purpose of the web site, which determines which techniques to use.

WAP Architecture

WAP Architecture defines two essential elements, an end-to-end application protocol and an application environment based on a browser. The application protocol, also know as the user agent is a communication stack embedded into each WAP device, the server deals with the other side of the protocol and communicates with the WAP client. The server must be split into two sections one which deals with normal HTTP request as used by all web servers, and the other deal with WAP request, this server is called the WAP gateway. The WAP device sends a request to the WAP gateway which translates the request to a HTTP request ready to be dealt with by the Web server, this simplifies the handling of both a WAP request and a HTTP request therefore the web server does not need to distinguish between a different requests. The WAP gateway has the capability of translating Web responses, that is a response from the web server into a WAP response ready to be delivered to the WAP device, this encoding and decoding of WML and HTML seems to be a powerful resource, however as HTML was never designed for small screen, low bandwidth devices. Therefore in most situations WML code is embedded into the server side language ready for delivery.

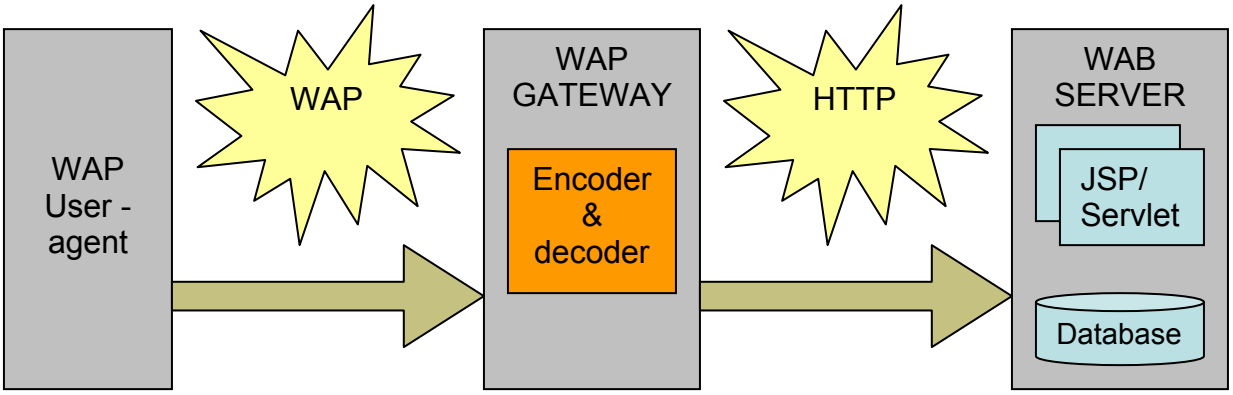


Figure 5 the WAP network structure

Which technology and technique to use.

The techniques to be used for personalisation will be explicit and implicit data collection, a form will prompt to user to input details and initialise a profile, and this will be done during Registration, and will ask for music preferences. The implicit collection of data will be carried out through session tracking and the uses of cookies. These techniques seem to be the most effective and accurate way of gathering data, the cookie method will allow for automatic logins, vastly improving the browsing experience of the user. The analysis method to be used will be rule based; this seems to be the simplest method giving high quality results, the algorithm used in filtering seems too complex without giving a large difference in quality.

New severed sided languages such as JSP and ASP have extreme advances over the traditional CGI, they are more efficient, convenient, powerful and portable than CGI. The main decision lies between ASP and JSP, although ASP is created by software giant Microsoft, ASP is restricted to only Microsoft machines and devices, this restriction in portability does not exist with JSP, because a JSP is a Servlet in its simplest form and these are simply Java classes, meaning JSP's and Servlets have access to the rich API created by Sun. More importantly they are platform independent, any device running the Java Virtual Machine can execute Java code, this may not seem so important with personalisation but becomes evidently significant when a Wireless devices is used. Because there are so many companies manufacturing mobile phones a language is needed that will allow execution across many devices, Java answers this question. But the main advantage Java over ASP is not of a technical issue but more of a business factor, the Java Community Process (JCP) is what gives Sun the overall advantage, created in 1995 the JCP is an open process devolved to revise Java technology, companies such as Nokia, Erickson and Oracle are some of the heavy weights members of the community and it is Nokia that developed Symbian. Symbain is an operating system for mobile phones written in Java, which dominates over 80% of mobile handset, giving it an extreme advantage through economy of scale. It is for these reasons Java seem the obvious choice and anyway why learn a new language when one knows Java.

The database to be used will be mySQL, no extensive research has been carried in this choice as the application is not meant to be data intensive, as it is using the rule based mechanism for personalisation, mySQL is also free to download and has good reviews so far.

The front end of the site will involve a number of different languages; computer accessing the site via Internet Explorer will receive Hyper Text Mark up Language (HTML), this is the set standard for Internet Explorer eXtensible Markup Language (XML) will also be used. For a WAP device Wireless Mark up Language (WML) will be sent to the device, finally a subset of Sun's Java 2 will be used to simulate a 3G device; Java 2 Micro Edition (J2ME) which is designed for small portable devices seems a good choice as Servlets and Java Server Pages are being used. Figure 5 illustrates the three-tier architecture with the given technologies.

Three tier architecture

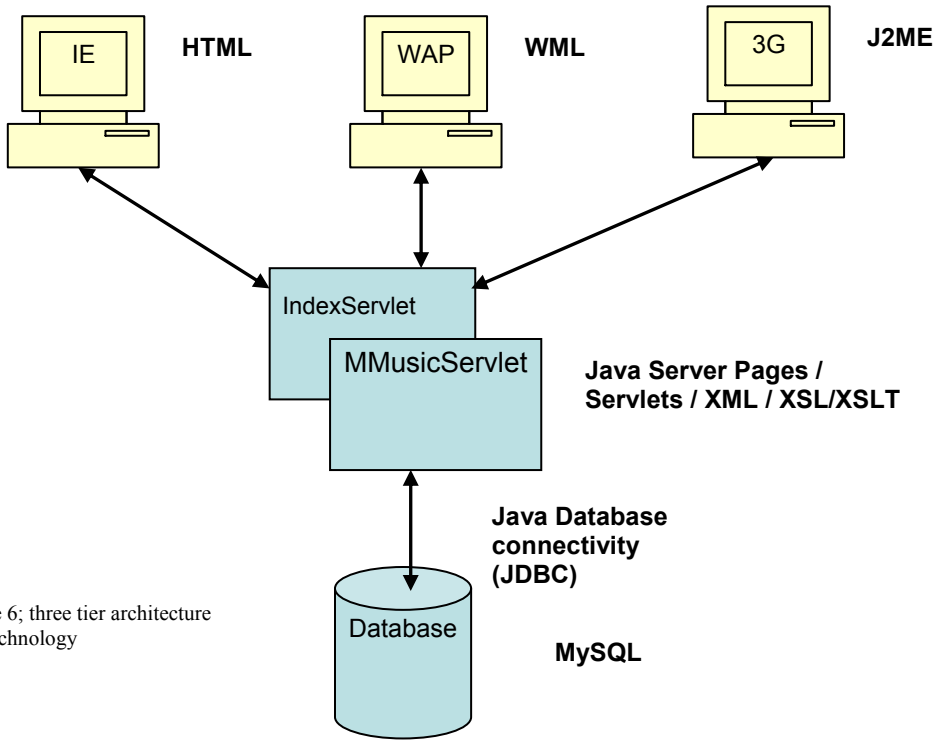


Figure 6; three tier architecture and technology

UML and the reason for choosing it

Devolved by the three "amigos" Booch, Jacobson and Rumbaugh, UML or Unified Modelling Language is the successor to a wave of object oriented analysis and design methods. It is most directly derived from Object Modelling Technique (OMT) but is standardised through the Object management group (OMG). It should be stated clearly that UML is not a methodology but rather the tolls used in a methodology, therefore UML can be described as a set of mainly graphical tools used to depict a system using object orientation.

UML has now become the de facto language for modelling object orientated systems and is widely used by most professionals, the exact reasoning behind why it has become so popular is largely unknown, however there are some considerable issues that help us to understand its popularity. Object orientation has become the new generation of languages and it seems to be growing ever more, UML is specifically designed to model objects and uses many of the concepts within object orientation, for example objects are instances of classes, UML provides class diagrams, these object can then have different states; UML provide state diagrams. A most desirable advantage of UML is its software life cycle model, as opposed to other techniques such as SSADM, which use the structured waterfall model, meaning an incremental, sequential method, without the option of back tracking or concurrently performing different stages. UML however is iterative and allows the different stages – requirements, analysis, design, implementation, testing and evaluation to be repeated and refined until they ultimately meets the overall system requirements. Software today just isn't made using strict waterfall method, in all but few exceptions each phase of the project breeds new issues and questions influencing the requirements.

One major factor which makes UML standout from any other method is that the creators of UML openly suggested that modification to the language be legit, they realised that for UML to remain the de facto standard an extension mechanism would be needed in order to keep up with vertical markets and rapid technological changes, therefore UML has been build in a controllable way which allows for extensions, without invalidating the language.

It is for these reasons that I have chosen UML as I feel knowledge of such a subject will allow me to express ideas and concepts more easily throughout the computing world.

UML Process considerations

Because UML is so flexible a number of different processes can be chosen or integrated, in the real world however the software processes becomes tailored for each application and a definitive process is not required. It is however good practice to identify a process or processes, which suit the application and create an outline of stages. Before an outline can be created a number of factors should be considered.

Makeup of the company and organisation, the type of process will depend largely on the size of the business, a large company with many pools of experience would require many different teams doing many different tasks, in such a case communication would be vital as artefacts would need to be complete and comprehensive in such a case a strict conducted process would hold the development together.

At the other end of the a small project team would require a less strict process, communication would be frequent and teams would have experience of working together, this does not however mean no process would be required, just that some aspects many play a less important role.

The nature of the application is another factor to consider, a human critical application would require a high levels of quality control, reviewing and testing would be a frequent activity and quality assurance would have strong emphasis.

Web application have a different prescriptive, it is unlikely that a web site would be critical to human life, it would however require intense input considerations as many different type of people would visit the site. Architectural factors would be especially important, as many levels of implementation may exist, if the application goal is to make use of new technologies strict process would not be beneficial as the project team would need to take advantage of discoveries. Security plays another vital role in web application especially commerce sites where payment transactions must be secure, great emphasis would be carried out in the design process to ensure this.

Skill level of development team, a team with low skill level normally requires a well-defined process as they many have a lack of experience.

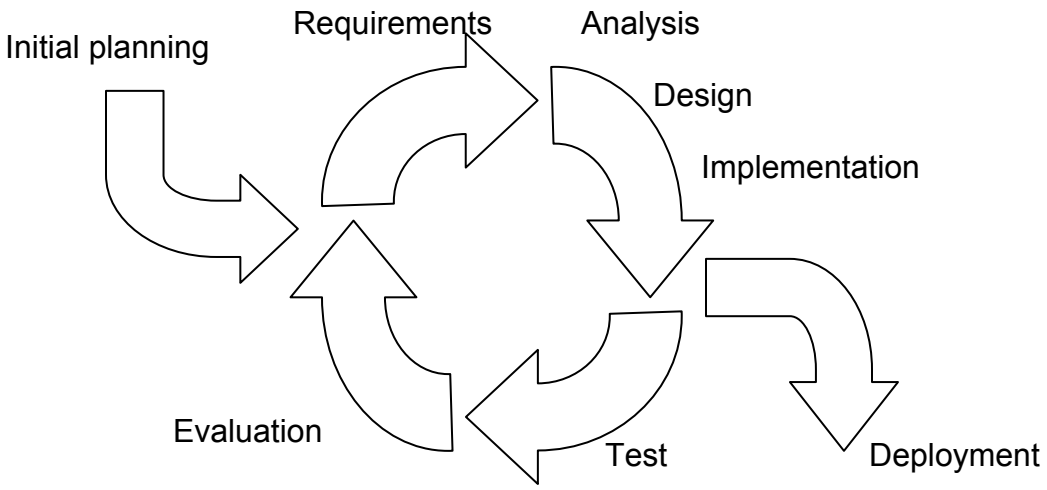


Figure 7. the iterative process

The overall process

As mentioned before UML allows the tailoring of processes, however one has neither great experience nor a deep understanding of UML or any UML processes, therefore some sort of process will need to be introduced, the most well know and well respected is the Rational Unified Process (RUP), this is very similar to most methods available but has key features which make it suitable for this project. As mentioned early the architecture of the system is important and will influence the design of the classes and structure of the objects, an architecture centric process such as RUP is appropriate, finally an iterative process is one that will help refine and strengthen the requirements, the incremental nature of RUP reduce the risks involved with inexperienced people such as myself. In addition to the RUP another process will be introduced; ICONIX Unified Processor or IUP, this introduces robustness analysis, by mean of robust diagrams. Robust analysis simple means analysing the user cases early on to identify classes and objects that will perform use case flow of events, identify attributes and behaviour associated with each of these classes and outlining the architectural mechanisms of the system. Robust diagram are useful as they will allow the implementation process to begin earlier, even if this is not the case at least the idea of implementation will become apparent earlier on, this fits in the nature of the project as a deadline has been set.

In addition to the RUP and IUP an extension will be added, this is legit as UML allows modifications; Web Application Extension WAE will be used in the design process to document web technology. An overall outline of the process and the products to be used is shown in Appendix C.

Section 4 – Development

This section was written once a more detailed view of the system was developed it documents changes in ideas and processes stated in the previous section. All in all in section gives a more accurate description of the final product and the techniques and technologies to be used.

The final process

Appendix C outlines the desired process to be used to develop an application such as the one in this project, however as stated in the main objectives, it is not desired to create a fully working commerce web-site, but rather to demonstrate the new technologies. For this reason many documents of the UML process have been omitted, rather than using UML to depict the detailed business logic of the application, which would require long periods of time and much effort in analysis, UML is used as a guide to the application and to define an abstract view. Therefore the following documents have been omitted; Uses Cases Analysis / Robustness Analysis, Package Hierarchy, Class Identification, Elaboration on sequence diagrams, Collaboration diagrams, Component diagrams, Refined Sequence diagram for web architecture, Logical View of classes, Component realisation, Link Relationship and Forms expression. All these documents are used to create a robust view of the system and to ensure no error have been made through the different stages. Uses Cases Analysis / Robustness Analysis for example is used to check robust sequence diagrams against the uses cases to ensure they both match up. Logical view of classes is used to provide detailed view of the classes and their interaction, this is of course an important step in any object oriented system and therefore a class diagram will be provided but detailed specification will not.

The new process is outline in Appendix D, and includes GUI prototyping, which is an effective way of rapidly getting an idea of the system, GUI prototyping gives “proof of concept” that is a physical view of what the system should do. It documents links form one page to another making up for the omission of Link Relationships step, input forms such as registration forms, are used in the prototype so parameters and methods which need to be handle by classes can be identified. The only downfall is backend class cannot be identified, this will require intuition and some experience.

Abstract definition of system

The abstract definition is a high level view used to define the overall aim of the system, although this stage has not been defined by either of the processes, I have chosen to include it to help myself get to grips with the main theme of this project. Appendix E step 1 defines two abstract definitions which I continuous refered to in order to keep the project running in the right direction.

Requirements specification.

The requirements specification is a collection of documents and models that attempts to unambiguously describe the system to be built. The documents included are the requirements of the system, model include use case and sequence diagrams.

The purpose of the requirements is to express behaviour or propriety the system should have, in general requirements should be categorised into functional and non-functional, requirements should be written clearly and understood by everyone who needs access to them, each requirement will be given an ID number used to refer back to it throughout the text.

The requirements document listed in Appendix E has been refined and is now the final document; the process of refinements came to me as the project progressed. As I decide earlier, cookies would be used to help track user sessions, I did not realise that cookies could be turn off by the user, I therefore needed to include requirement 2.1, which dealt with the scenario that cookies have been disenabled.

Requirement 4 was also added as I discovered that small amounts of data could be added to cookies, this meant I could store user name and password details, which meant the user could automatically login.

Use Cases

Use cases describe a set of cases that are likely to occur in the system, they do not presume any design or implementation factors, actors are used to describe generic roles of a user or external system. One interesting point in the uses case is the construction of use cases, traditionally two relationships are used; includes and extends. Includes involves one use case making complete uses of another, the extends relationship allows one use case to extend from another which could be executed in certain condition. Although this method is widely used in many UML texts I prefer the idea put forward in Open Modeling Language (OML) as documented by Rosenberg,

Rosenberg, D 1999, Use Case Driven Modelling with UML, Addison Wesley 0-201-43289-7

Who uses invokes and proceeds, invokes is used much like a function call when a use case is called by another case. Proceeds indicate one use case must precede another within a sequence. As long as I use this method throughout my design, I cannot foresee any problems no.

The use case model consists of a general diagram and individual use case scenarios based on the use case definitions. The general model gives an overall view of cases however this diagram can be somewhat ambiguous as it does not show clearly how different use case will be executed; the individual diagrams allow us to get a clear picture of the cases and the line of execution to be accomplish by these cases. As we will see this is be beneficial during the sequence diagrams and more importantly during testing.

Looking at the use case general diagram we can see that no reference to wireless or wired actors, the reason for this is that I assume same functionality will be executed for both wired and wireless clients, the cases will be the same but the content (method of displaying case options) will be different. Anonymous and non anonymous customers are used as actors as the system will have to execute difference functionality for the different customers, a definition can be found in Appendix B.

A further look at the use case diagram gives rise to a interesting use case the larger “cookie check” case, this seems the most complex of all the cases, it is likely that the class for this case will also be complex. I should imagine this class will include the “check browser” case and will form the backbone of the system. Much care and attention will be required when implementing this class.

Sequence Diagrams

Sequence diagram are dynamic models, which aim to flesh out logical view of the uses cases, they help us to form a sequence of actions which need to be carried out in order to complete a use case. The boxes at the top help us to get a idea of the classes in the system, the arrows and text give us an idea of the functionality to be carried out to complete a use cases.

The sequence diagrams in step 2.4 make no attempt to indicate any classes, at this stage there is no clear web architecture set out, therefore I have no knowledge of HOW the system will execute use cases. I do however know the logical steps which need to be carried out in order to complete a use case; this knowledge comes from common sense, as we can see in the diagrams a very basic outline is give on how to complete a use case. There is no indication to a certain language and the only indication of technology is Hyper Text Transfer Protocol (HTTP) request. It is quite clear that a HTTP request will have to be used, as most web architecture is bases on this protocol. Therefore the sequence diagrams in step 2.4 gives us an abstract view of the steps in sequence to be carried out.

Most of the diagrams in step 2.4 seem fairly simple, the most complex case is UC1 which again includes “cookie check” as many checks need to be performed on the client, for this reason I should attempt to spend more of my resources on building classes for this use case.

GUI Prototyping

GUI prototyping allows you to get a feel of the entire system very easily and with little ambiguity, unlike sequence diagrams GUI prototyping gives us a look the front end of the system and more importantly how web pages with link together. This helps us build a picture of how classes will interact, it is likely for every web page there will be a class handling input in the background, a link from one web page to another also means a link from one class to another. Prototyping also helps us get an idea of the attribute and methods in a class, for instance a input field in a web page would require a variable in the class that handles it, this variable may need to be inserted into a database which would need some method to do so.

Class Identification

This step involves looking at all the nouns in previous documents and deciding whether they are classes or not. Intuition is needed to identify classes this way as no outline or rules are given in helping finding classes. As we can see form step 3.2.1 the classes found are;

- Registration
- Cookie
- Login
- Profile
- Customer

Because only a few classes have been found no attempt to create a class diagram is made, for the simple fact that there just isn't enough to go on.

Web application extension

The WAE is used to define exactly how the system will work, before we can do so an explanation of the technologies being used must be given, after which a set of extensions can be defined and used to model the system.

Java 2 Enterprise Edition (J2EE)

J2EE is a three layered system designed to meet the requirements of dynamic industry; it is used widely by companies requiring rapid application development with complex business logic. An example of this could be a banking system, where records of customer transaction need to be kept accurate and update at all times over the web. Traditionally programmers used Servlets to deal with the business logic, but soon found this to be too tedious and were not able to meet deadlines set by organisations, for this Sun Microsystems introduced Enterprise Java Beans or EJB. EJB deals with this problem by allowing programmer to use an API designed for enterprise logic this helps build application rapidly, also the nature of a Java Bean allows much complex transactions to be accomplished more easily.

EJB therefore deals with complex business logic and sits at the bottom of the three layers; Servlets are used to deal with more simple functions such as user name verification and navigation through the web site, they do however operate on the web server, by definition a Servlet is a Server Element. Java Server Pages or JSPs were introduced to help produce web content more rapidly JSP in their real form are Java Servlets, but rather than embedding HTML, or any other mark-up language into the programming code JSP embed programming code into the mark-up file. This mean a JSP can deal with data without first sending it to a Servlet. As a general rule JSP should be used when a lot of mark-up language is needed and Servlets are used when more functionality is needed. A diagram showing J2EE layers is show in figure 8.

This project will use the first two layers of J2EE, as no complex operations are needed, EJB therefore will not be used. Servlet technology will be used to execute all functionality by the system and JSP will by used to show dynamic web pages.

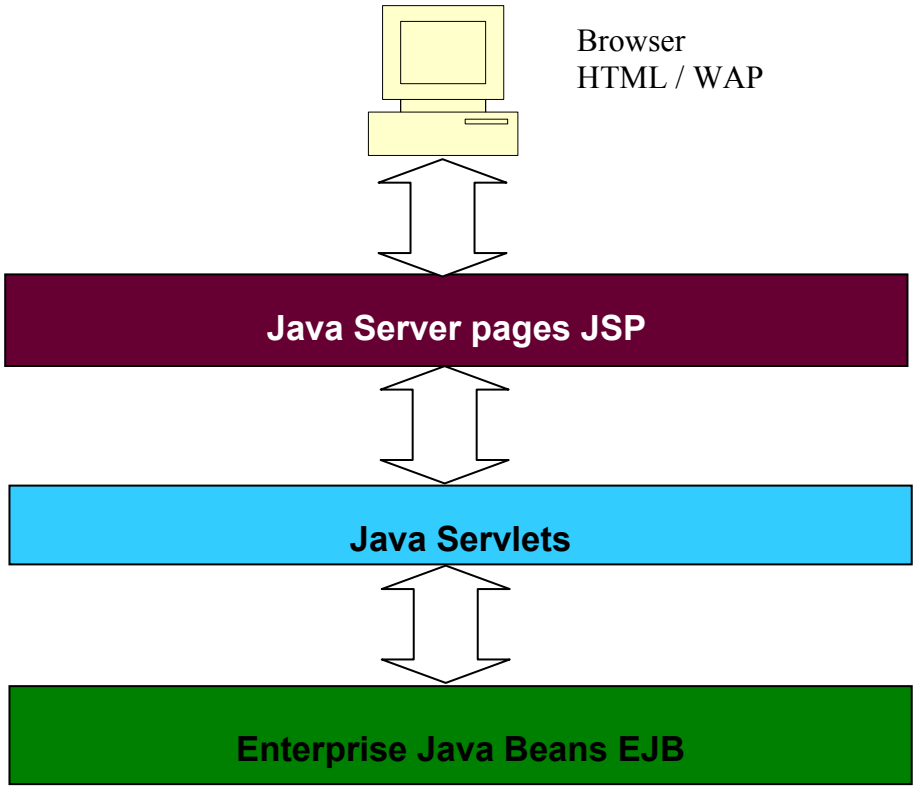


Figure 8 J2EE architecture.

Servlet Technology

A higher-level view of Servlet technology shows that they work as Humans do when we communicate, when a person asks another person a question they wait for an answer, this could also be seen as one person *requesting* an answer for another person, and then waiting for a *response.* Servlets too require a request from the user, they then perform computation and deliver a response, this simple method forms the backbone of Servlets and most other server sided technologies. All data needed by a Servlet is passed around the system in a `HttpServletRequest request` object and anything that needs to be sent to the client is stored in a `HttpServletResponse response` object. The problem with this method is that each time control is passed from one Servlet to another the request and response objects are reset.

There are two request types in Servlets GET and POST both are used to call Servlets from any mark-up language, GET is the most simplest to implement however it has added security risks as the data is written into the URL, therefore sensitive information such as passwords can easily be seen by others, there is also a limit to the amount of data which can be sent as it is written in the URL. POST is a more secure method and also allows unlimited data size but area little more difficult to implement.

As mentioned above request and response object have flaws as they are reset as control is passed around the system, a method is needed to allow a object to remain active while the user browsers, this come in the form of `HttpSession session`. This object remains alive while the user has their browser open; important information is stored here such as products put into a shopping cart. Any data can by stored in a session object, but it is important to assign one attribute to the session to allow other Servlets to identify the session.

Java Server Pages

JSP in essence are Servlets, they are just used when a lot of content and less functionality is needed, JSP also have request and response objects, they also have a session object however these are built into the JSP and are hardwired once the JSP is complied to a Servlet. There is no need to include these object into your code.

Java Beans

A Java Bean can be defined as a class that holds data, for each variable a set and get method is defined and some functionality is included in the bean. Bean form the basis of object orientated programming

Apache Tomcat

Apache Tomcat is the actual web server used to execute the Servlets and JSPs. It handle the http request sent by the user and finds the correct Servlet to execute, it also complies JSPs into Servlets automatically, all the Servlet API classes are kept inside Tomcat. Written under the Jakarta project by Apache, Tomcat is the most popular of all web Servers that support Java, other server include Microsoft's IIS and W3C's Jigsaw, but as Tomcat is written in pure Java and Apache has such close links to Sun Microsystems and Java, it seemed the most suitable choice. (Even the project name is linked to Java, Jakarta is the capital city of the island Java.)

XML

XML is simply text delimited with special characters like HTML, the difference is HTML has predefined meaning set for each tag, when a browser reads the <table> tag is creates a table on the screen. In XML there are no predefined tags, programmers are free to create their tags which they define in the Data Type Definition (DTD), XML therefore has no intrinsic meaning and this is wear the power of XML comes into its own. The purpose therefore is to hold information in a form that is suitable for any application, XML provides a mean of storing data in elements that mean exactly what you want them to mean. Because these elements have no meaning, it is platform independent which makes it a desirable technology for the future, as new devices and language become part of the internet XML will allow them to communicate. The question now is how we get elements within a XML file into a format that is desirable, there are many answers to this question a program could read the data within an XML file and create an object based on the data, another method is to use a style sheet or a XSL (eXtensible Stylesheet Language) file which sets out the rules to transform the XML file.

XSL

XSL defines a set of instructions to transform a XML document, the basic structure of a XSL file is first to identify a XML tag and then to perform some function. For example a tag defined as <title> in an XML document would be transformed into a <H1> tag in HTML, the XSL file would first identify the <title> tag and then output a <H1> tag with the information contained in the <title> tag.

XSLT

The actual transformation of an XML document is performed by a XSLT engine (eXtensible Stylesheet Language Transformation), this engine is written in some 3rd generation language which takes a XML and XSL document as input and outputs a document defined by the XSL document. XSLT can be found in development programs simulating output or can be used as a background program such as a Servlet. Before any transformation can take place, a XML parser must first parse the XML file. An XML

parser puts the XML document into a state that allows functionality to be performed on each tag. There are currently two methods of parsing Document Object Model (DOM) and Simple API for XML (SAX). DOM creates a tree like structure, putting each tag into a tree node; this allows operation to be performed on any node of the tree, meaning more complex functionality can be performed, the pitfalls however is that the entire document needs to be stored in memory and the implementation of DOM is more complex. SAX works like a procedural programming, the tag are put into a list like structure and operation are perform to blocks of the list as they came along, SAX is less dynamic and complex, and some complicated operations are more difficult to perform, however a new technology, eXtensible Query Language or XQL is emerging which allows you to query any block within the list like structure alloing more complex operation to be performed.

One can see a resemblance to the Data Structure unit (QC204) where binary search trees and linked lists were introduced. Figure 10 outline the DOM and SAX methods with an example, we can see that the DOM method is more efficient, consider a scenario where it is was required to perform functionality on the <p> tag, the DOM method would only require the traversal of one node, (<body>), SAX would require a massive six nodes.

Writing a XML parser and XSLT engine is way beyond the scopes of this project, this level of programming is done by large organisation and academic organisations, instead I will use versions given as freeware by companies.

The current range of parsers and transformers available for download are; James Clarks XP and XT, IBM Corp. XML4J Apache Xerces, Microsoft Corp. MSXML, Oracle Corp. XML Parser for Java, and Sun Microsystems Java Project X. It is quite clear that Microsoft's MSXML would be an inappropriate choice as it is not written in Java, as are the rest, and tests show it performs the worst out of all. Sun Microsystems Java Project X is proven to perform the best with 100% accuracy, however the only version that are open source are IBM Corp. XML4J Apache Xerces and James Clarks XP and XT. Although Apache version would be a adequate choice as I am using Apache Tomcat, James Clarks XP and XT gives 95% accuracy and is currently the fastest parser and transformer on the market. Therefore I will use James Clarks XP and XT engines in my project. Figure 11 outlines the XSLT process.

Ahmad Abualsamid, Network Computing, http://www.networkcomputing.com/1106/1106f23.html
http://webreference.com/xml/column22/2.html
http://www.jclark.com

Web application extension

Now that we have outlines the technologies a clearer picture can be formed of how the system will work. We know that for each page i.e. Index or Login page, a XML file will need to be created, and a XSL file for each device which corresponds to the XML file. Figure 12 shows a specfic view of this process. We also know a XSLT engine class is needed to perform the transformation, James Clark uses Servlet technoloy to creat XP and XT engine, so we will need to introduce a Servlet extenstion. The output

of the XSLT engine will be any form, in our case we know it to be HTML or WML, so some XSLT output will need to be defined. JSPs will also be used to form some of the pages which cannot be done by the XSLT process, and lastly a session object which is of the upmost importance if we are to allow tacking to take place. In addition customer detials will need to be held dynamically while the session is alive, this will require a basic Java Bean. Appendix E step 4.2 defines the web extension for UML.

DOM and SAX data structures

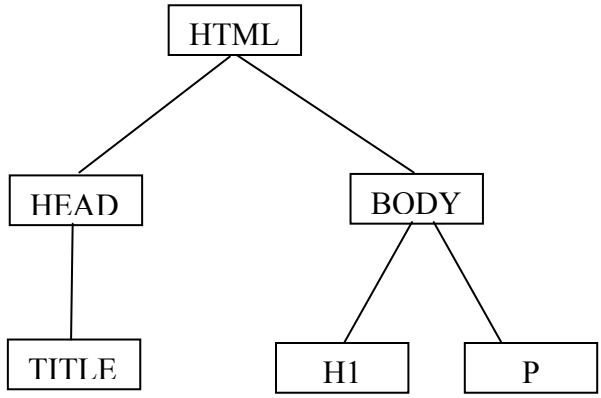
HTML source extract from www.jclark.com

```
<HTML>

<HEAD>
<TITLE>James Clark's Home Page</TITLE>
</HEAD>

<BODY>
<H1>James Clark's Home Page</H1>
<P>Information about the following is available:</P>
</BODY>
```

DOM method, tree structure



SAX method, list structure

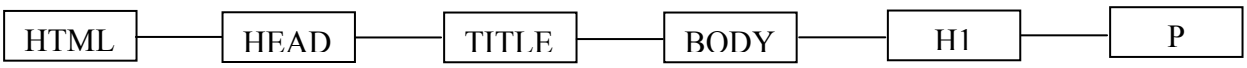


Figure 10 DOM vs. SAX

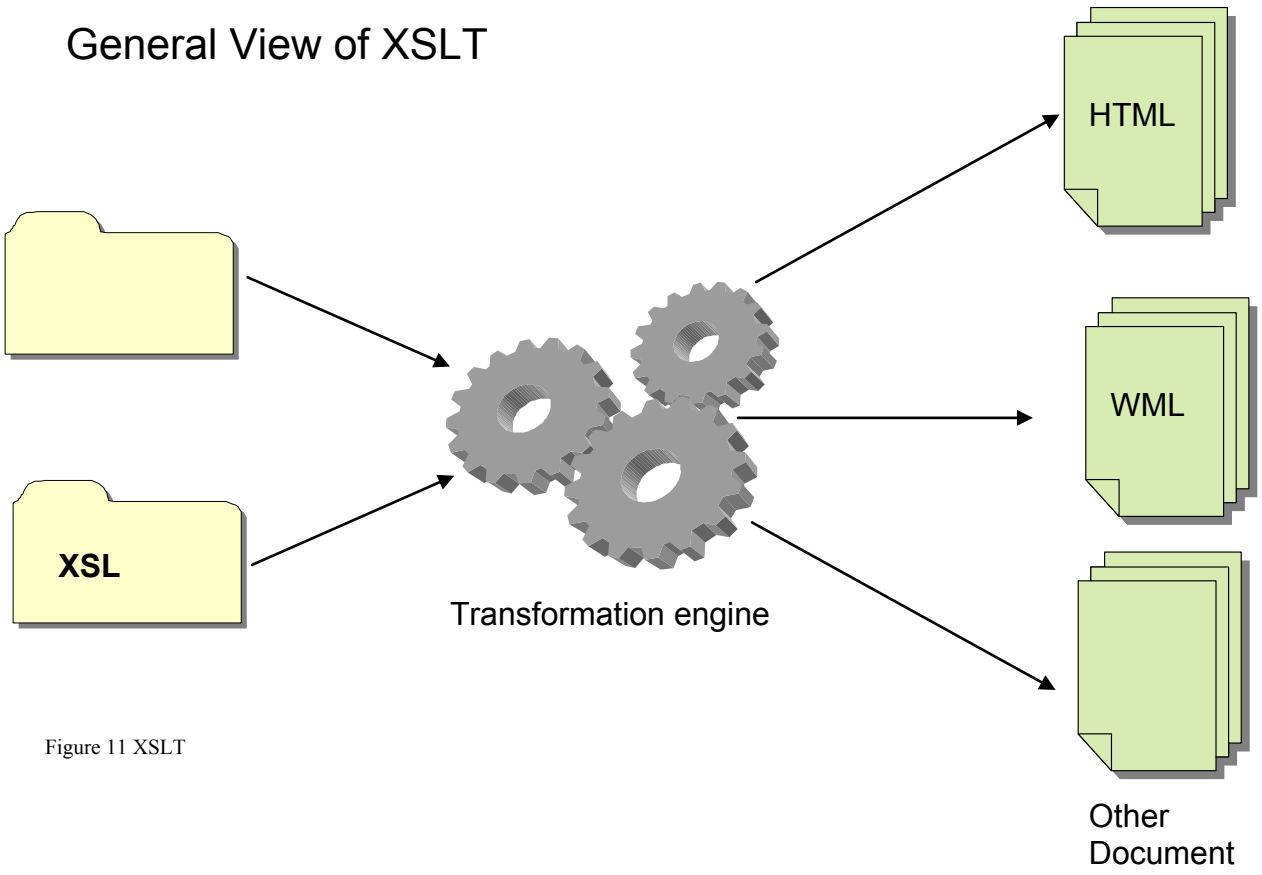


Figure 11 XSLT

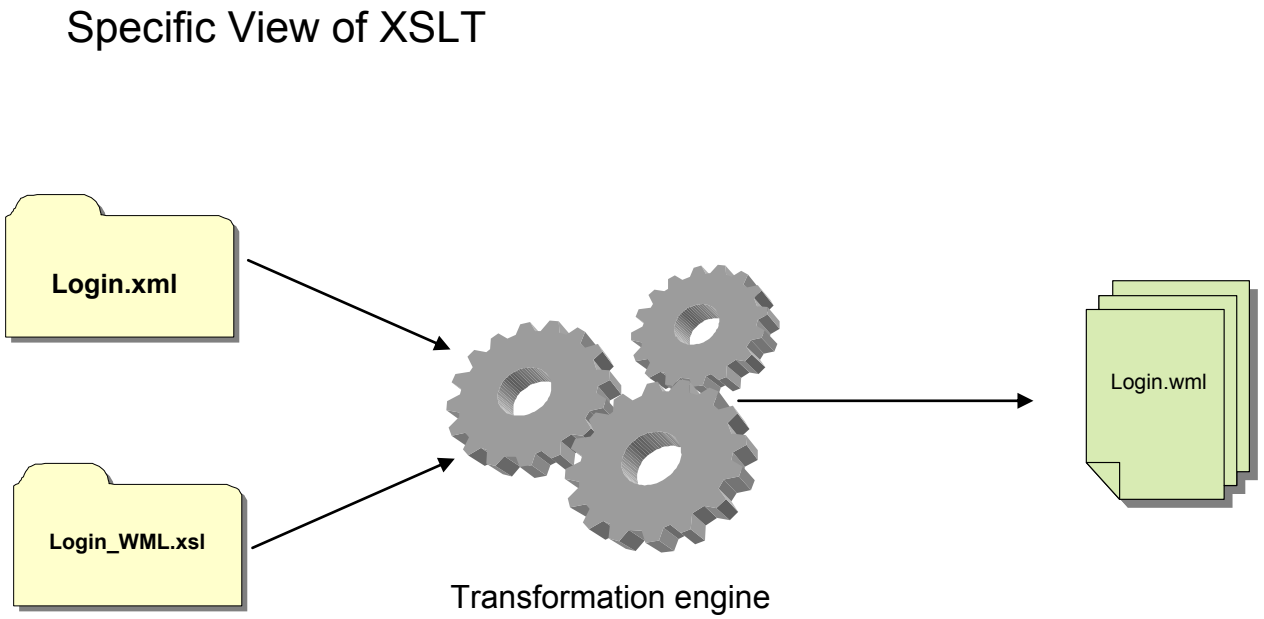


Figure 12 Login XSLT example for WAP

Section 5 - Implementation

Documentation

Note: A clear distinction must be made before the documentation begins. Index.jsp is a JSP that redirects control to the main Servlet Controller; it is not a displayable page in itself, just a file that can be access by a WAP or PC device, the main index page which displays a welcome message is created on the fly using XML and XSL. Any reference to index page is aimed at the welcome page created on the fly.

Servlets

Package controller

Package controller can be seen as classes which control data coming in from the user and sends data and information to the user.

Client controller – Authenticator.java

The Authenticator class is used to identify two thing, firstly whether the user has a session and secondly whether the user has a cookie which has been deployed by M-music. Many text books define methods for check whether a user has a session, which is to set an attribute to the session and then check whether that attribute exist, this is normally define as a String attribute with no real meaning, perhaps the name of the company or web site. If this attribute exists the user has a live session else a session must be created. Authenticator also uses this method, the difference is the type of attribute used, instead of choosing any name I decided to assign the clients browser type, since this information is vital to the system and since the session object remains live while the client device is connected to the system, it seem a chose that would accomplish a important task whilst saving memory in the session object. Why assign two variables to the session object one to identify the session and another to record the client browser type.

The method of detecting user client is implemented in class Detector.java which is a Servlet, Authenticator extends Detector and therefore inherits all its method. The method `detectClient( request, response )` is used to determine the client browser type.

Authenticator handles four scenarios; the first is a user has no session and no cookies, in this case it is the user first visit to the web site and a session is created and Index page is displayed. If the user has no session but cookies are available it means the user has visited the web site before, cookies were deployed with user name and password. In this case a session is created, the client type is detected and assigned to the session, the user name and password are retrieved from the cookie and assigned to session object, control is then passes to Login Servlet. In the next two scenarios a session object has been created; in this case the user is browsing through the website and has made some request, because the request is only know at runtime the Servlet waits for a parameter to be sent from the user, which is embedded, in the Mark-up files. We can see the this class related to the “cookie checker” use case, it also integrates the “check browser” case. Figure 13 shows the general view of the Authenticator class.

```
If user has no session and no cookies
     Then  Get Client type, create session and display index
page.
If user has no session and has cookies
     Then  get  Client  type,  create  session  and  login
automatically
If user has session
     Get users request display requested page
```

Figure 13 Authenticator class general view.

Device Container – Device.java

The only purpose of this class is to hold data on the device types. In this case only two devices are used; the PC device is given value 1 and is referred to as Device.PC, the WAP device is given value 2 and is referred to as Device.WAP.

Page Container – Page.java

Like the Device class Page only hold data on the pages to be shown, Index is give value 1 and is referred to as Page.INDEX, Login 2 and is referred to a Page.LOGIN, Registration 3 referred to as Page.REGISTRATION and music page is 4 referred to as Page.MUSIC.

Client Detector – Detector.java

The Detector classes main task is to detect the client type, this is done by reading two attributes from the HTTP header found in the request object, for a HTML browser the HTTP header has the value “mozilla” for a WAP browser the header is "text/vnd.wap.wml". A test is made using the is the indexOf() method, if the browser is HTML a variable is set to 1 or Device.PC else if the browser is WAP then the variable is set to 2 or Device.WAP. This is assigned to the session object in the Authenticator.

Request Controller – Location.java

If Authenticator class is seen as the brains of the system Location.java is the backbone, any request made by the user will eventually invoke this class, it handles which XML and XSL documents to be sent to the XSLT engine. Firstly the class establishes the request made by the user which is stored in the request object and then determines the browser type held in the session object. These two tests determine which XML and XSL file to be sent to the engine, for example a request to view the Login page by a WAP user would send Login.xml and Login_WML.xsl.

Package handler

Package handler is used to handle data coming in to the system, the data is not changed in any way and is normally inserted in the database. Classes Registration and Login sit directly behind a mark-up page waiting for some parameters to be sent, these classes both check whether a session has been created, if one has not it immediately send control to the Authenticator class which deals with the problem.

Registration Servlet – Registration.java

Registration has a number of variables which are assigned values from the user; this is then used to create a Customer bean which is assigned to the session object. An attempt to deploy cookies is made at the end, the Profile page is displayed by sending control to the Request Controller Location class with parameter Page.MUSIC.

Login Servlet – Login.java

Login can be invoked in two ways firstly by the Authenticator when cookies have been detected, in which case Login retrieves the user name and password from the session object. The other way is when the customer makes a request to Login via the login mark-up page. In both case the login detail are verified by making a connection to the database, once this is completed a Customer bean is create and assigned to the session object. The profile page is displayed in the same way as the Registration Servlet. One important fact to note is when Login is invoked by the Authenticator the user name and password are stored in the session object, this means that information is duplicated as user name and password are also stored in the Customer bean. The session object needs to be kept “tidy” and free of redundant data, much like a rational

database, for this reason the user name and password are erased from the session object once a Customer bean has been created.

Customer Bean – Customer.java

The Customer bean is used to hold all the customers data, including the six music types, it has a connection to the database and defines three methods `initCustomer(), insertCustomer()` and `updateCustomer()`. The first method is used to initialise all the Customer variables, the method is called by the Registration class and Login class. The `insertCustomer()method` insert the customer variables into the database. The update method is used when a user changes their profile.

Personalised Music Page – Profile.java

Because the Profile music page, which display all the customer music preferences and their name is created dynamically XML and XSL could not be used, instead a JSP is used, this is defined below. Profile Servlet sits directly behind the Profile JSP and handles the user's music request. The JSP hardwires parameters for each request, if the user clicks on the first music choice button the value 1 is sent to Profile Servlet, the Servlet then checks the Customer bean for the users first music type and hands control to the XSLT engine with the correct XML and XSL files. Profile Servlet acts in a similar way to Location except it deals only with music requests.

Although this method is sufficient, I would rather let Location Servlet deal with the music XML and XSL files; this follows the rules of reuse and allows easier maintenance. Profile should only work out which music type to send to the Location Servlet, in addition a Music Container should be defined stating all music types as Device and Page class do.

Change Customer Profile – ChangeProfile.java

This class is not been defined, the aim was to take all the new customer details and call the `updateCustomer()` method in the Customer bean.

*Package com.jclark.xsl.sax.**

This package is defined by James Clark and uses the SAX method of parsing, this seem to be adequate as I was not performing any real complex operations. A problem a raised when I was trying to send XML and XSL file to the main Servlet. I soon found that this along with many other classes would need to be re-configured, a task that proved to be of up most difficulty, I took nearly a week to get a picture of the package structure. The following three classes were altered.

James Clarks –

XSLServlet.java, ServletDestination.java and XMLOutputHandler.java

XSLServlet.java is defined as the main Servlet in the package, the problem was the Servlet was only taking file which were hardwired into the code, because I need my file to be sent as parameter I needed to change the Servlet. Two paths were created one virtual path which defined the path of the XML and XSL file form the tomcat directory, and then I added a real path which defines the whole path of each file until the tomcat directory Because all the XML and XSL file are kept in the same directory only the name of the file was needed, this is attached to the request object when control was pass to the engine in the Location Servlet. The XSLServlet now uses a file protocol in the format file:// to access the files.

This allowed me to view HTML pages transformed by the engine, but the engine was not outputting WML, I kept getting “page not allowed” errors on my WAP simulators. I soon discovered that the problem was the Content type. The Content type is a header sent to the device telling it what the content is, WAP uses “text/vnd.wap.wml." Somewhere in the code the content type was set to the HTML standard, I knew this because the WAP simulators shows the content type. I discovered this was hardwired in the ServletDestination class and handled in the XMLOutputHandler’s init method, which took a Destination and AttributeList object as its parameters. The Content type needed to be sent in a Destination object and because ServletDestination is a child of Destination the content type could be set there and then sent to the init method of XMLOutputHandler, two methods were added in the ServletDestination class getContentType and setContentType. A ServletDestination object was created in the XSLServlet file which set the correct content type and the object was then sent to class OutputMethodHandlerImpl, which handled it as it would normally. When control was finally passed to XMLOutputHandler, the process of which is beyond my knowledge, ServletDesination was type casted to a Destination object and content type retrived the calling the getContentType method. This was then assigned to the String variable previously defined. Admittedly this configuration had an element of luck; I was not always sure about what I was doing and nervously back up many copies of the original files.

Java Server Pages

Personalised Music Page – Profile.jsp

This page dynamically creates content in the form of six buttons in accordance to the Customer profile defined in the Customer bean. The JSP get the music types and assign the names to the buttons, it also displays the users name.

Change Customer Profile – ChangeProfile.jsp

ChangeProfile JSP is not created, the aim was to create a page similar to registration, and the JSP would then get all the customer details from the bean and add them to the fields in the page.

XML and XSL Documents

It is not the aim in this section to go through all my XML and XSL documents, instead to give a general view of the techniques used to integrate the two document types effectively and efficiently. As discussed before the XML document is used to hold all possible information to be displayed to the user, the problem is how to separate data that the PC device can see and that which only a WAP device can see. I got around this problem by creating two types for each information holding tag, the first would have a simple name such as “general”, which would hold the bear minimum amount of information which would be useful to the user, the second tag would be called “general_extra” this would include extra information which could be displayed on a larger device. So when the XSL document for a PC device would execute with its XML file it would display both “general”and “general_extra”, the XSL file for WAP would only execute the “general” tag.

Section 6 – Testing

The testing phase will be based on the Use Cases, which is know as uses case based testing in the software engineering world. Two set of test will occur one for the WAP device and one for the PC device. The findings can be found in Appendix G.

The results for the PC device are all passed, however I did not fine the time to complete the WAP side 100%, I know from the results of the PC device the backend java logic is working, but I did not setup the links between WML pages and Servlets for the WAP device. In other words I have no completed integration of the whole system. Instead I will show results of the XML and XSL file which could be seen as modular testing, this is accomplished by using a program called XMLSpy which includes a built in XSLT engine. Appendix G documents this.

Section 7 – Critical Assessment

Although much work and research has been done in this project there are many issues that would be changed if I could redo the project, firstly a grave mistake was not setting up a development environment, I just went straight ahead once I had an idea of my system and began coding. What I should have done is tested James Clarks XSLT engine, got it working and then start coding. Another mistake was not to clearly document my modules as I went along, I did write an account of them but what I didn't do was to write an account of all the data flowing in and out of the modules. This meant when I came to integration many of the attribute names, which were used by many modules were different, I feel this increase integration by two fold. This also holds true for my XML and XSL documents, many of the parameter which were sent to the Servlet had different names across the different files.

If we look at the Customer profile their music preference are repeated for example my profile reads House, Techno, Soul, Techno, House and Funk, this is down to the rule base method in my database. The rules are as follows;

Music	Similar 1	Similar 2
House	Soul	Techno
Techno	House	Funk
Soul	House	Funk
Funk	House	Soul
Latin	Soul	Funk
Trance	Hard House	Techno
Hard House	Trance	Techno
Big Beat	Trance	Techno

We can see in my instance House and Techno were my first and second choice, and we can see House will display Soul and Techno, Techno will display House and Funk. There is repetition; House, Techno, Soul, *House*, Funk and *Techno.* To avoid this an algorithm could have been used in the program to identify repetition music and pull off different similarities.

Looking at the content we see that most of the pages are non-dynamic, I could have got around this by using Java Applet, which would have been embedded in the XML, XSL file and would run off the server, by definition an applet is a Application Element or an element which would sit inside the application.

I was not able to use J2ME, as it is not used to display web site content, J2ME is used to build mobile applications such as games.

Section 8 – Conclusion

This project has pushed my thinking near to its limits due to the complexity and lack of time, however I have gain experience and knowledge which I am sure will benefit me in the future. Regarding the standard of this project I feel disappointed that I could not complete the system in time, but I know the system will be completed at some point in the near future. Much has been covered in this report but there is still so much I could have included, this is because the mobile industry is being shaped now as you read this and new ideas and technologies are being introduce all the time, the possibilities are only restricted by our imaginations! – As Albert Einstein said “Imagination is Greater Than Knowledge".

Bibliography

Jakarta Project homepage - http://jakarta.apache.org/

IPV6 homepage - http://www.ipv6.org/

Java Cookie API - http://java.sun.com/products/servlet/2.1/api/javax.servlet.http.Cookie.html

Developer Fusion homepage- http://www.developerfusion.com/

Sun Java homepage - http://java.sun.com/

Sun wireless java homepage - http://wireless.java.sun.com/

Wireless Marketing - http://www.marketingsource.com/articles/viewall/67

M-commerce article -http://www.ecommercetimes.com/

Personalisation - http://www.clickz.com/res/personal/index.php/all

Forum Nokia developer page - http://www.forum.nokia.com/main.html

http://training.gbdirect.co.uk/courses/mobile_computing/

http://www-106.ibm.com/developerworks/library/xml-perl2/

http://www.xml.com/mobile/

http://www.w3schools.com/xsl/xsl_functions.asp

Professional WAP Arehart C, Wrox Publishing

Deitel Java How to Program Prentice Hall

Deitel Wireless Internet and Mobile Business Prentice Hall

Hall M Core Servelts and Java Server Pages Sun microSystems

Ben Forta WAP developement and WML and WML Scrpit Sams

Tuner MySQL and JSP web Applications SAMS

Conallen J Building Web Appliactions with UML Addison Weley

Fowler M UML Distilled Addison Weley

Rosenberg D Use Case Driven Object Modeling with UML Addison Weley

Appendix A

Appendix B

Definitions

Definitions

Anonymous customer - Customer who is unknown at connections time, they may be registered, but they do not have any cookie data available to be processed by the system.

Non anonymous customer – Customer who is known at connections time, they have registered, and cookie data is available.

Mark-up page – Any page written in a Mark-up language is this project it is likely to refer to HTML, WML or both.

Appendix C

Methodology Outline

1. Abstract definition of system.
   1.1. Abstract definition of system

2. Requirements specification.
   - 2.1. Requirements Gathering.
   - 2.2. Uses Cases.
   - 2.3. Uses Cases Model.
   - 2.4. Sequence diagram.
   - 2.5. Uses Cases Analysis / Robustness Analysis.

3. Analysis
   - 3.1. Package Hierarchy
   - 3.2. Class Identification and diagrams
     - 3.2.1. CRC Cards
     - 3.2.2. Role Playing
     - 3.2.3. Noun identification
   - 3.3. Elaboration on sequence diagrams
   - 3.4. Collaboration diagrams.

4. Design
   - 4.1. Component diagrams.
   - 4.2. Introducing WAE and Client Web architecture.
   - 4.3. Sequence diagram for web architecture.
   - 4.4. Refined Sequence diagram for web architecture.
   - 4.5 Logical View of classes.
   - 4.6. Component realisation.
   - 4.7. Link Relationship.
   - 4.8. Forms expression.

Appendix D

Methodology Outline Final

1. Abstract definition of system.
   1.1. Abstract definition of system

2 Requirements specification.
   2.1. Requirements Gathering.
   2.2. Uses Cases Model.
   2.3. Uses Cases.
   2.4. Sequence diagram.

3. Analysis
   3.1 GUI prototyping
   3.2 Class diagram
      3.2.1 Noun identification

4. Design
   4.1. Web application extension.
   4.2. Sequence diagram for web architecture.

Appendix E

UML Methodology

1. Abstract definition of the system.

1.1 Abstract definition of the system.

ID: AF1. The system should allow wired and wireless clients access to the web site.
ID: AF2. The system should display personalised information.

2. Requirements Specification.

2.1 Requirements

Functional

Requirements ID: R1
Title: The system shall be accessible via wired and wireless clients.
Wireless client should have a WAP browser with WML capabilities.
Wired client should have a web browser with HTML capabilities.

Requirements ID: R2
Title: The System should deploy cookies onto the client's hard disk documenting the number of visits and login details.
2.1 The system should prompt registration or login for clients that have disenabled cookies.
2.2 The system should request users using a WAP device to access a wired device to register

Requirements ID: R3
Title: The System should prompt user registration on the third visit to the web site.

Requirements ID: R4
Title: The system shall allow non anonymous customer to login automatically via their cookie data.

Requirements ID: R5
Title: The system shall allow online users to input music interests via registration and store the information on the server database.

Requirements ID: R6
Title: The system shall display the user's music interests randomly and automatically once logged in.

Requirements ID: R7
Title: The system should allow users to change their music profile.

2.2 Uses Cases.

Access to home page.

ID: UC1

Goals:
Online customer wishes to view web index page.

Brief Description:
An anonymous customer wishes to view the website through a wired device; a HTML or WAP enable browser is needed to display the maximum information. The web site should then deploy a cookie to the hard disk documenting; one visit has occurred.

Main success scenario.
Customer accesses web site via HTML or WAP browser.
The system checks for available cookies on customers hard disk.
The system deploys a cookie onto the hard disk stating that one visit has been made.

Extension.
2a: Cookie is present on the user's hard disk.
2.a.1: If a cookie is present the visit count is incremented by one.
2.a.2: The system then checks the visit variable within the cookie, if it is greater than three and registration is not complete the system prompts for registration.

3.a: Cookies are disenabled on users hard disk.
3.a.1: The system prompt user to register.

Pre conditions
HTML or WAP compliant browser
The customer web browser should be capable of supporting HTML or WML and HTTP requests.

Priority: High

Risk: Low

Requirements: RD1, RD2 RD3.

Customer logins.

ID: UC2

Goal: Customer logins in.

Brief description:
An anonymous customer logins in via the homepage link. The user should enter their email address and password correctly in order to successfully login, once they are login they may view their personal page. If they have forgotten their password or entered it incorrectly a forgotten password page is displayed.

Main success scenario.
Customer accesses logs in successfully.
System displays personalised page.

Extension.
1.a: A forgotten password page is displayed.

Pre conditions:
Browser type has been detected.

Priority: Med

Risk: High

Requirements: RD1.

Customer registers.

ID: UD3

Goal: Allow anonymous customer to register.

Brief description:
An anonymous customer enter their registration details via the registration link on the home page, they are also asked to choice their music preference. Once this is done their personal home page is displayed. If they re-type their password incorrectly a message is displayed.

Main success scenario:
A customer has successfully registered and the system has successfully documented this in the cookie.
Customer views personalised information.
Customer updates profile.

Extension
1.a Re-type password is incorrect, display error.
1.b Cookie disenabled, display error, proceed to personal page.

Priority: med

Risk: med

Pre condition:
Browser type detected

Requirements: R5 R6.

Automatic login.

ID: UC4

Goal:

Brief description:
A non anonymous customer automatically logins via their cookie data and their personalised page is display automatically.

Main success scenario:
Customer login automatically into the system.
Customer views personalised menus.

Extension:
1.a
1.a.1 Email sent to customer

Priority: med

Risk: High

Pre condition:
Browser type detected

Requirements: R4 R6.

Change profile.

ID: UC5

Goal:

Brief description:
A customer viewing their personalised page changed their profile and views their new personalised page..

Main success scenario:
Customer changes profile.
Customer views new personalised menus.

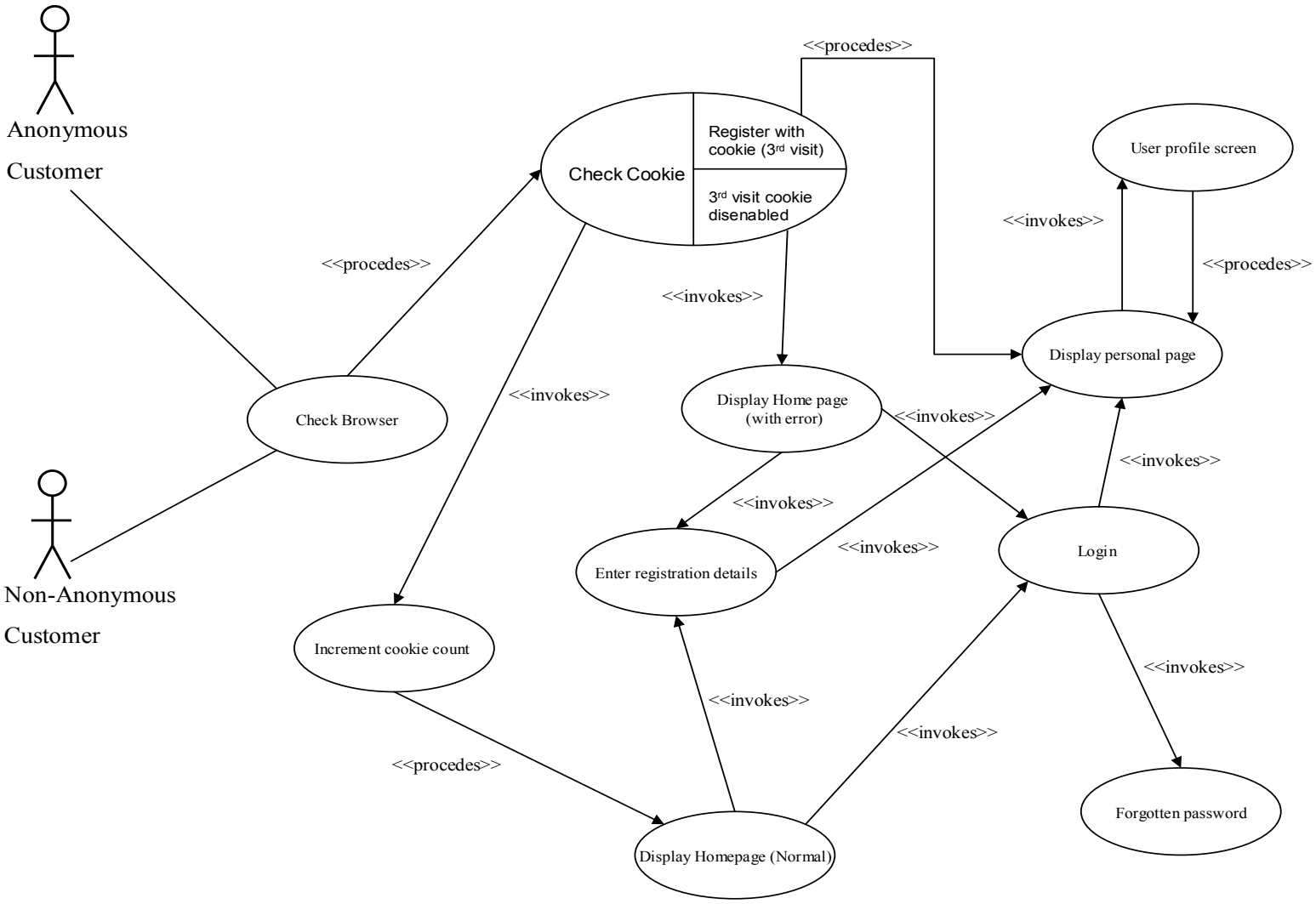
Priority: med

Risk: Low

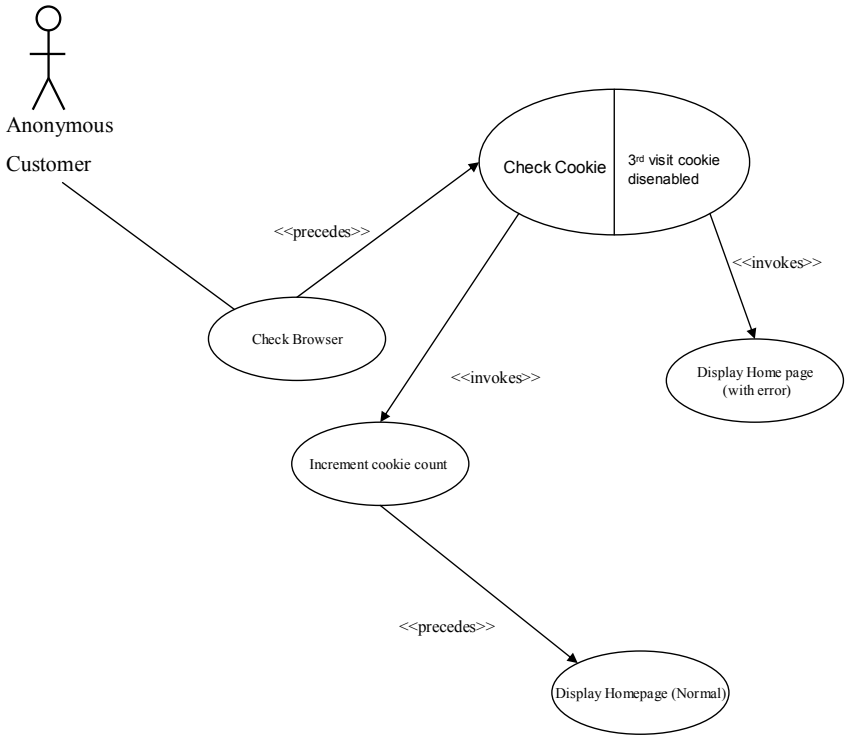
Pre condition:
Browser type detected

Requirements:R7.

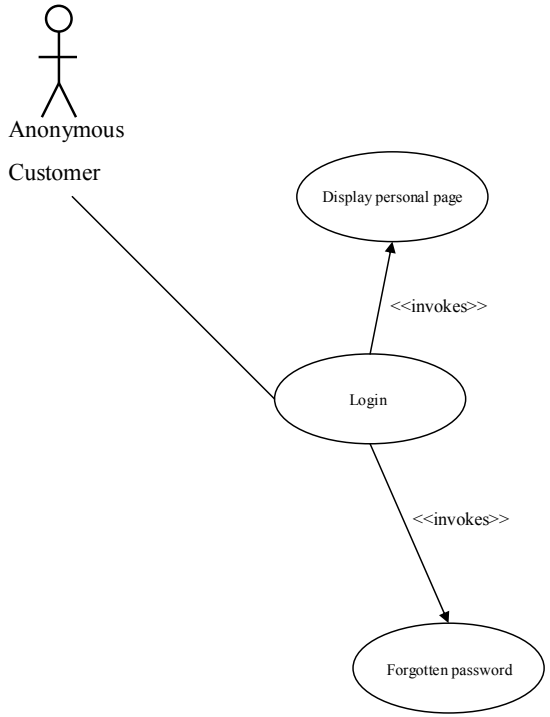
2.3 Use Case Model -
General Diagram



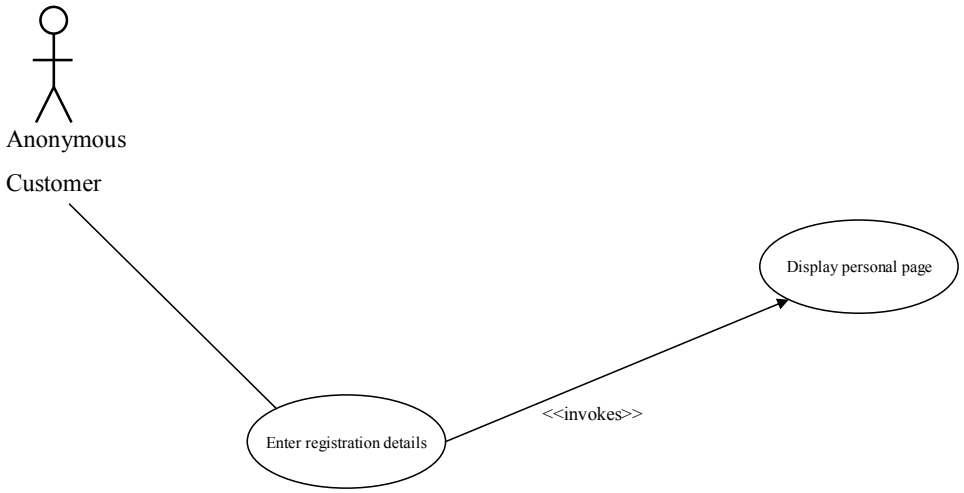
Use case Scenario1: UC1 - Access to home page.



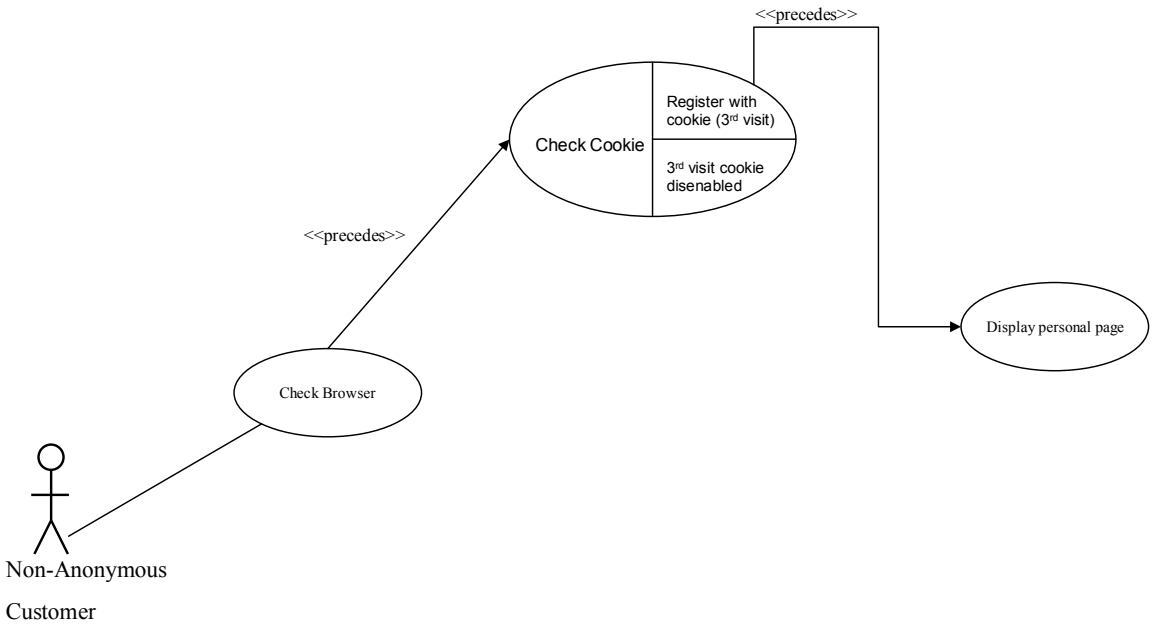
Use case Scenario 2 : UC2 - Customer logins.



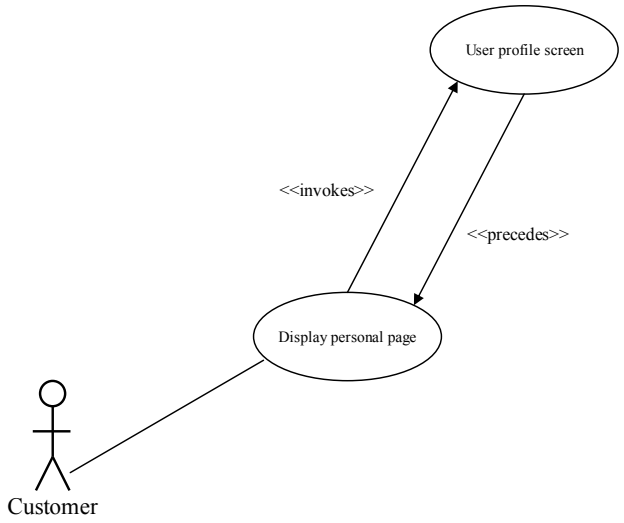
Use case Scenario 3 : UC3 - Customer registers.



Use case Scenario 4 : UC4 - Automatic login.

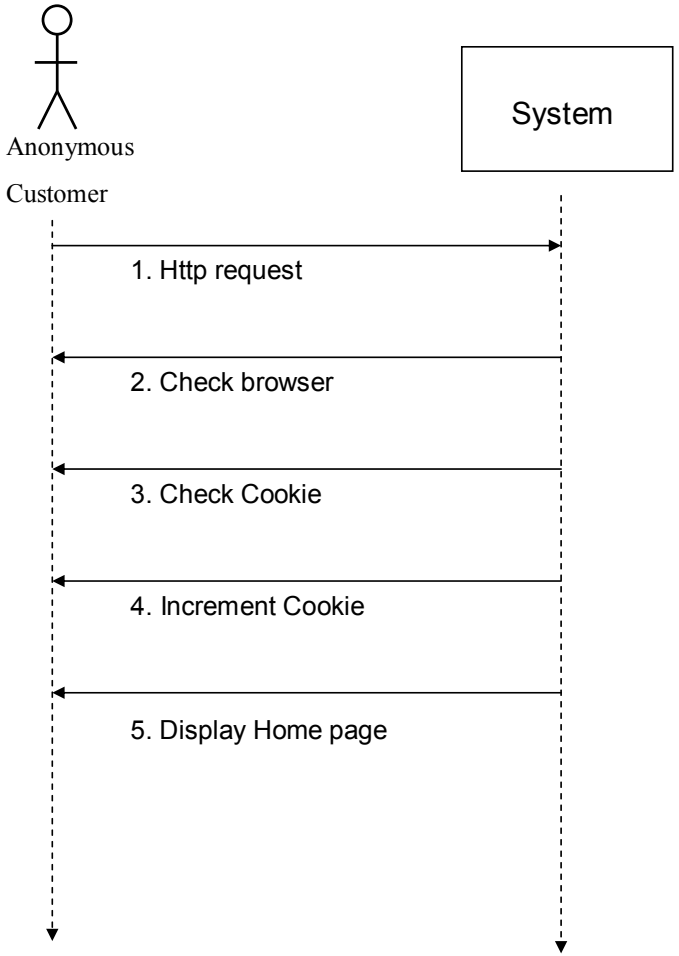


Use case Scenario 5 : UC5 - Change profile.

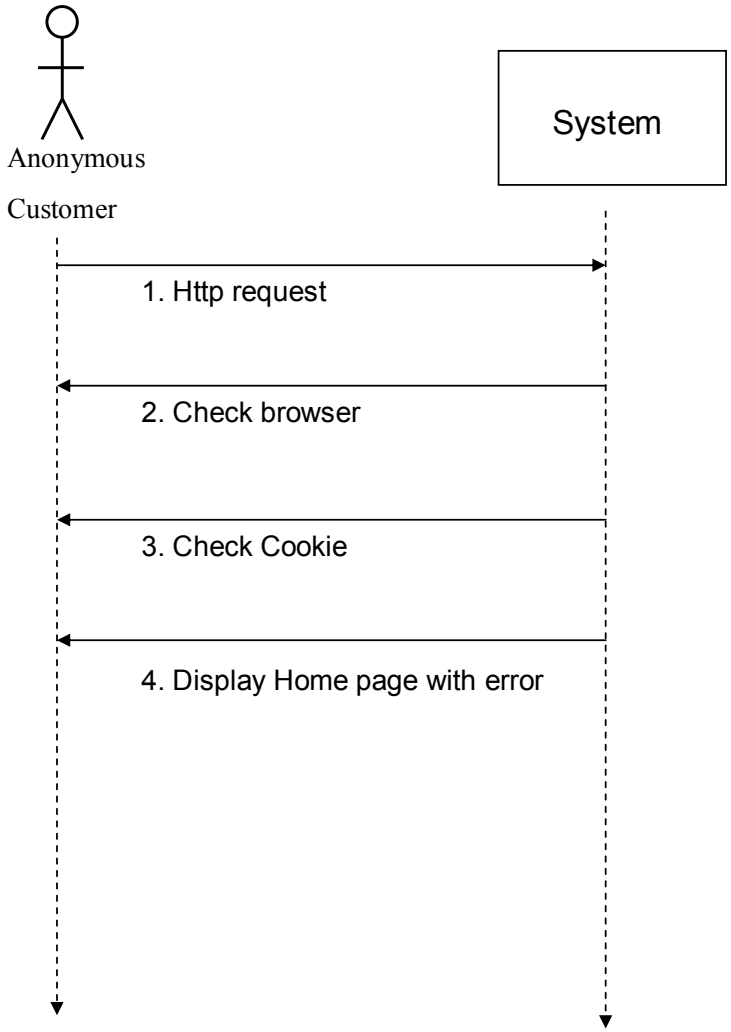


2.4 Sequence Diagram -

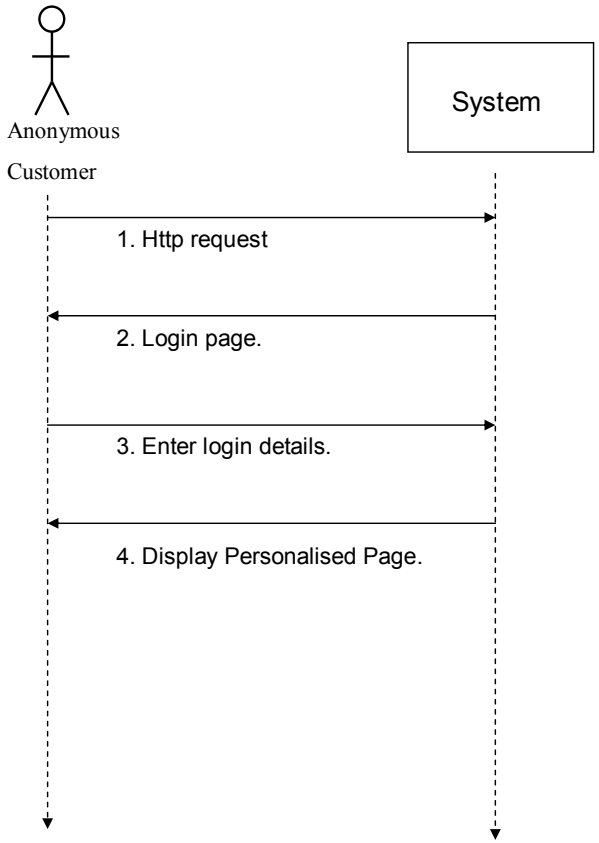
Main success scenario for UC1



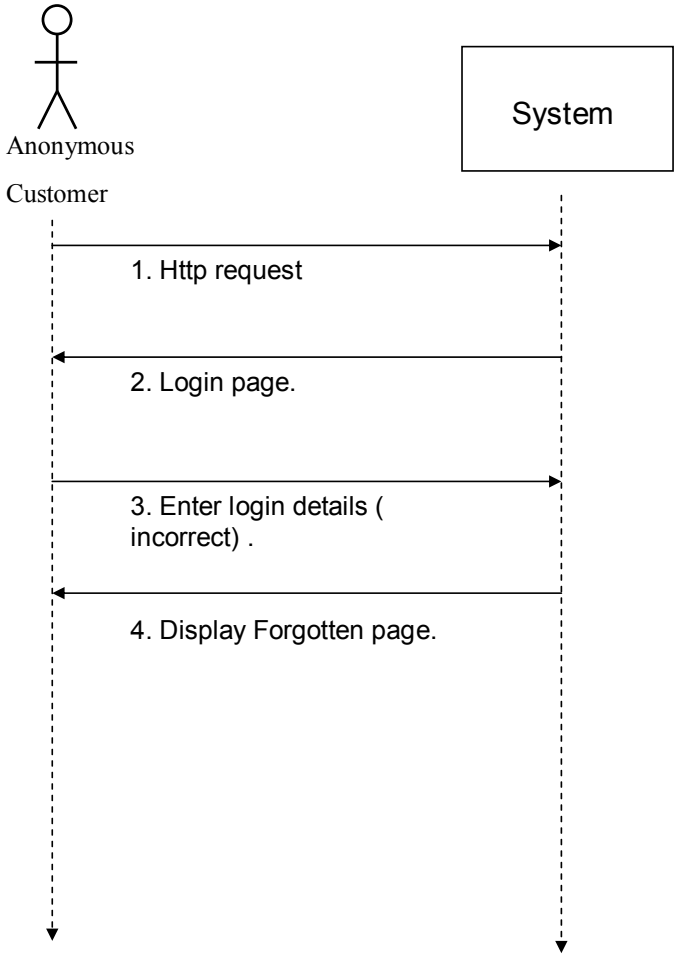
Cookies unavailable scenario for UC1



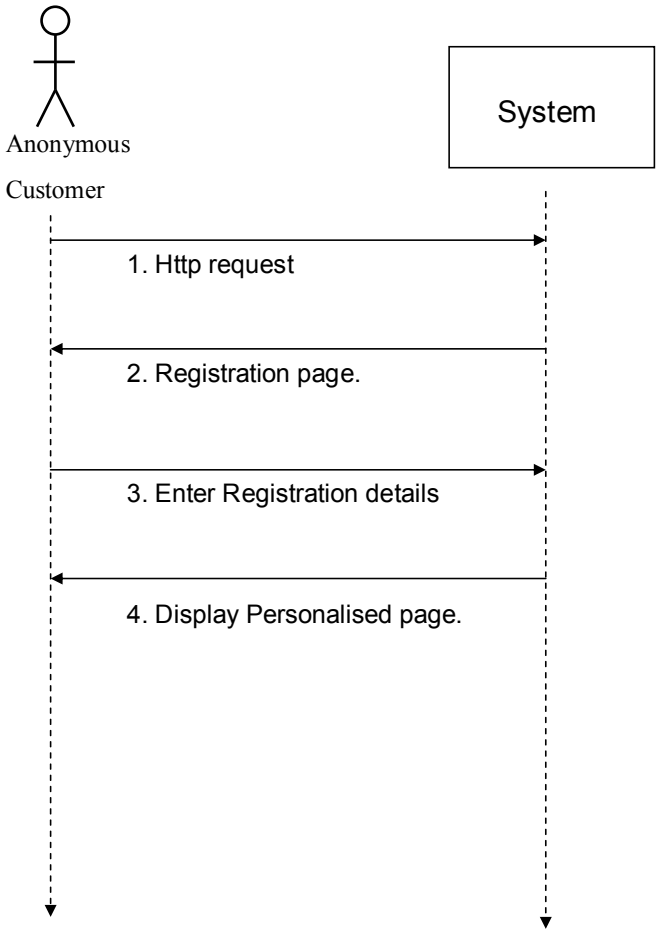
Main success scenario for UC2



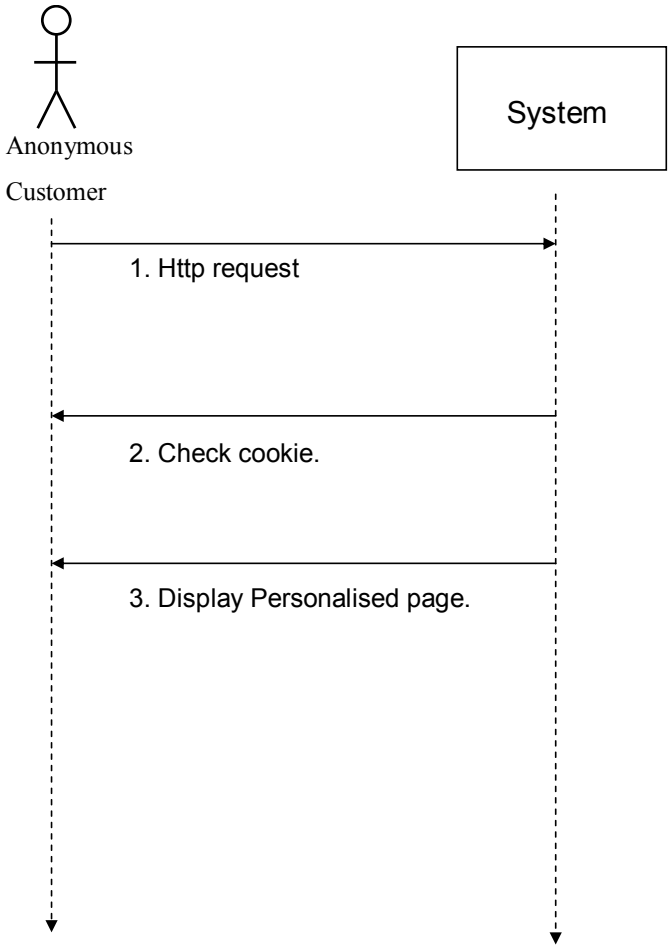
Forgotten password success scenario for UC2



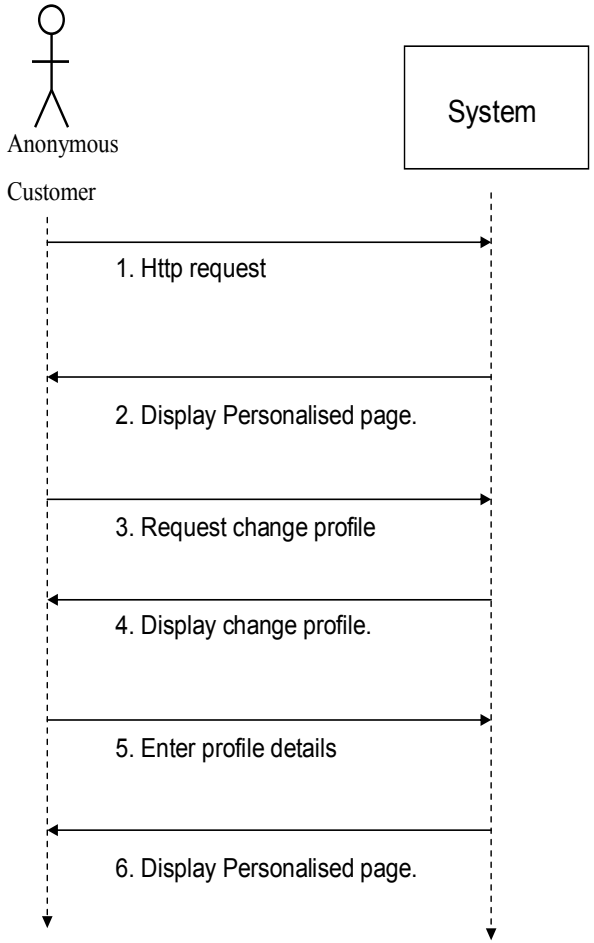
Main success scenario for UC3



Main success scenario for UC4



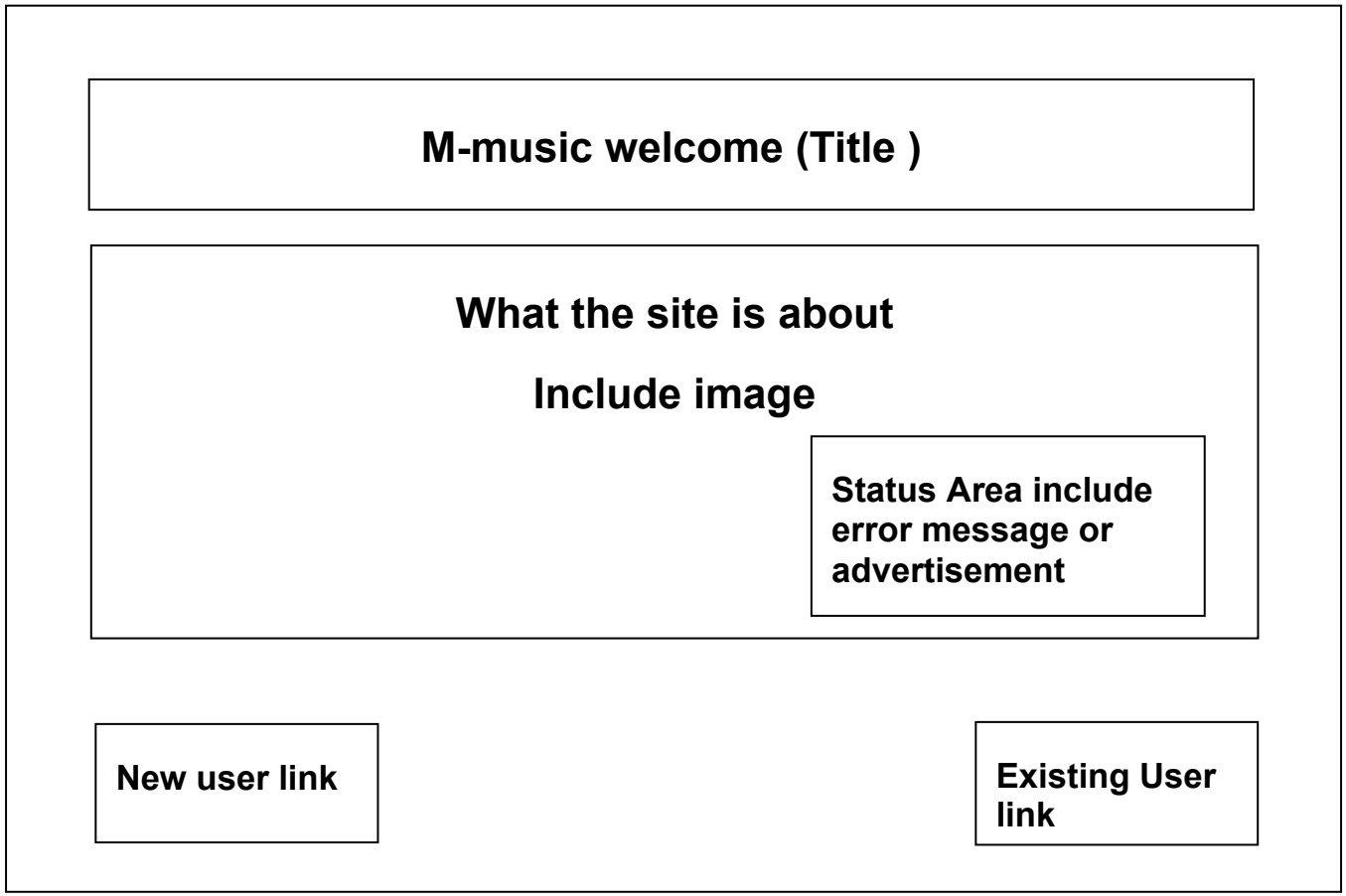
Main success scenario for UC5



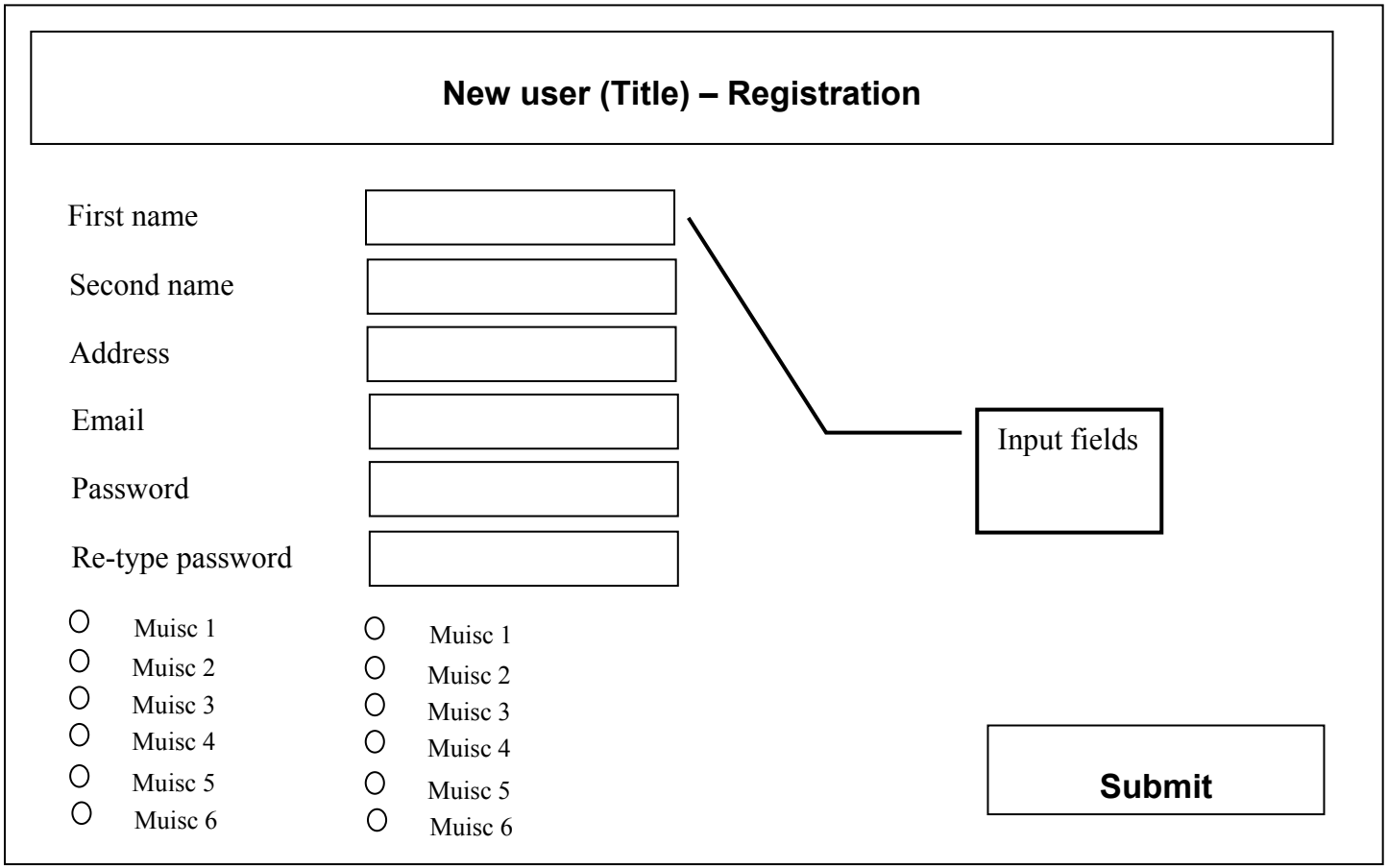
3. Analysis

3.1 GUI Prototyping –

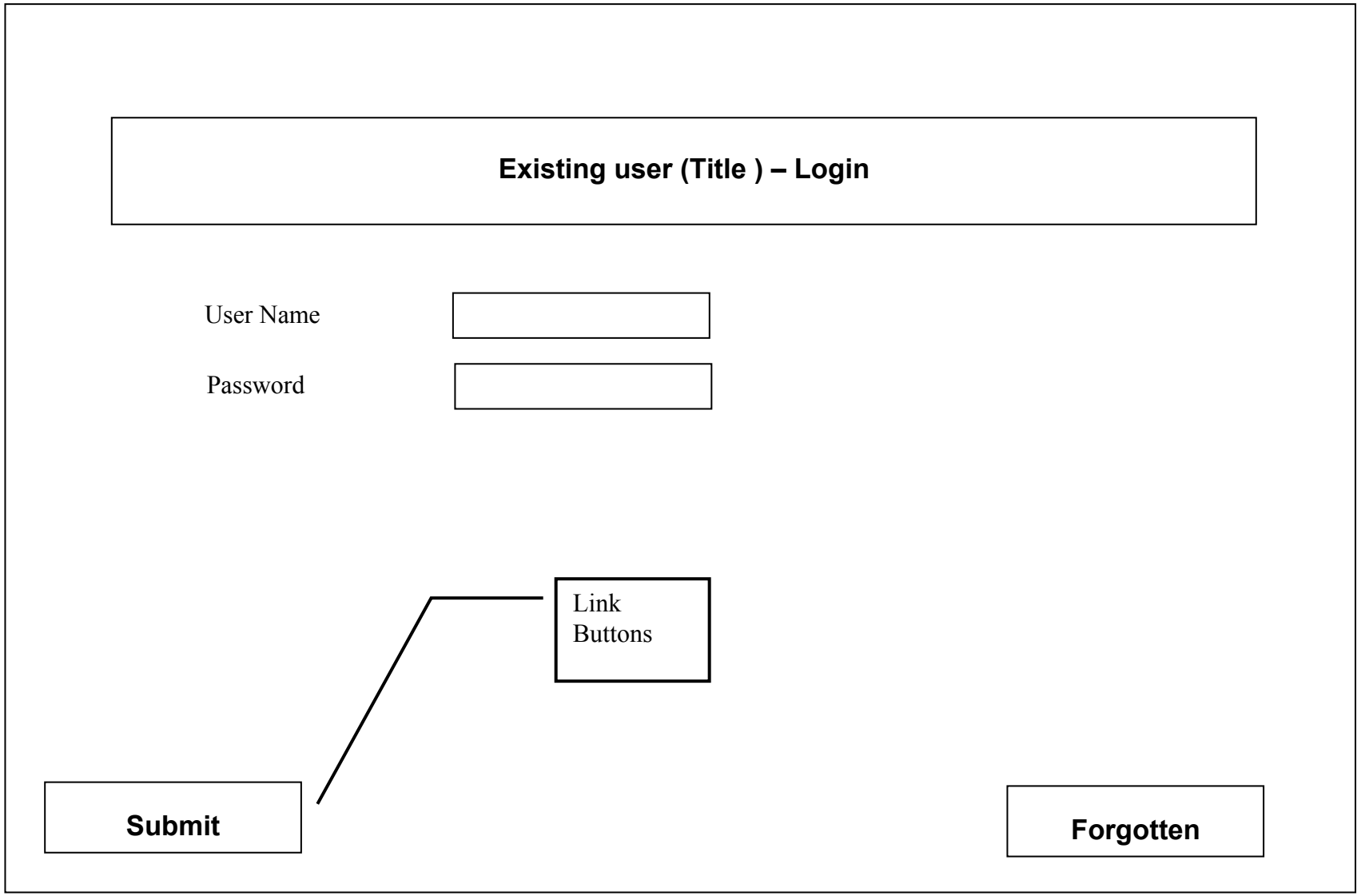
Wired HTML Browser Prototype – Homepage



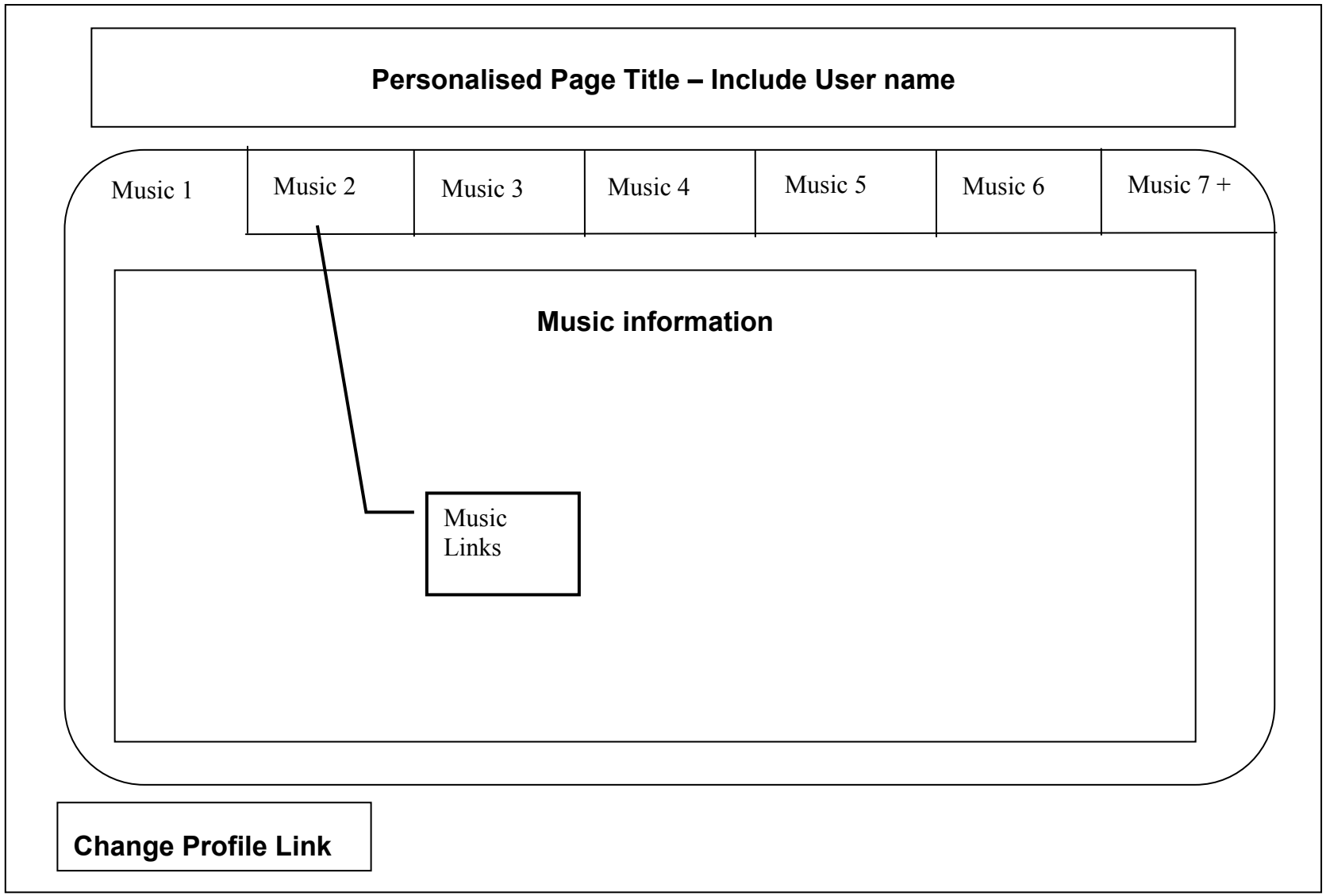
New user (via new user link)



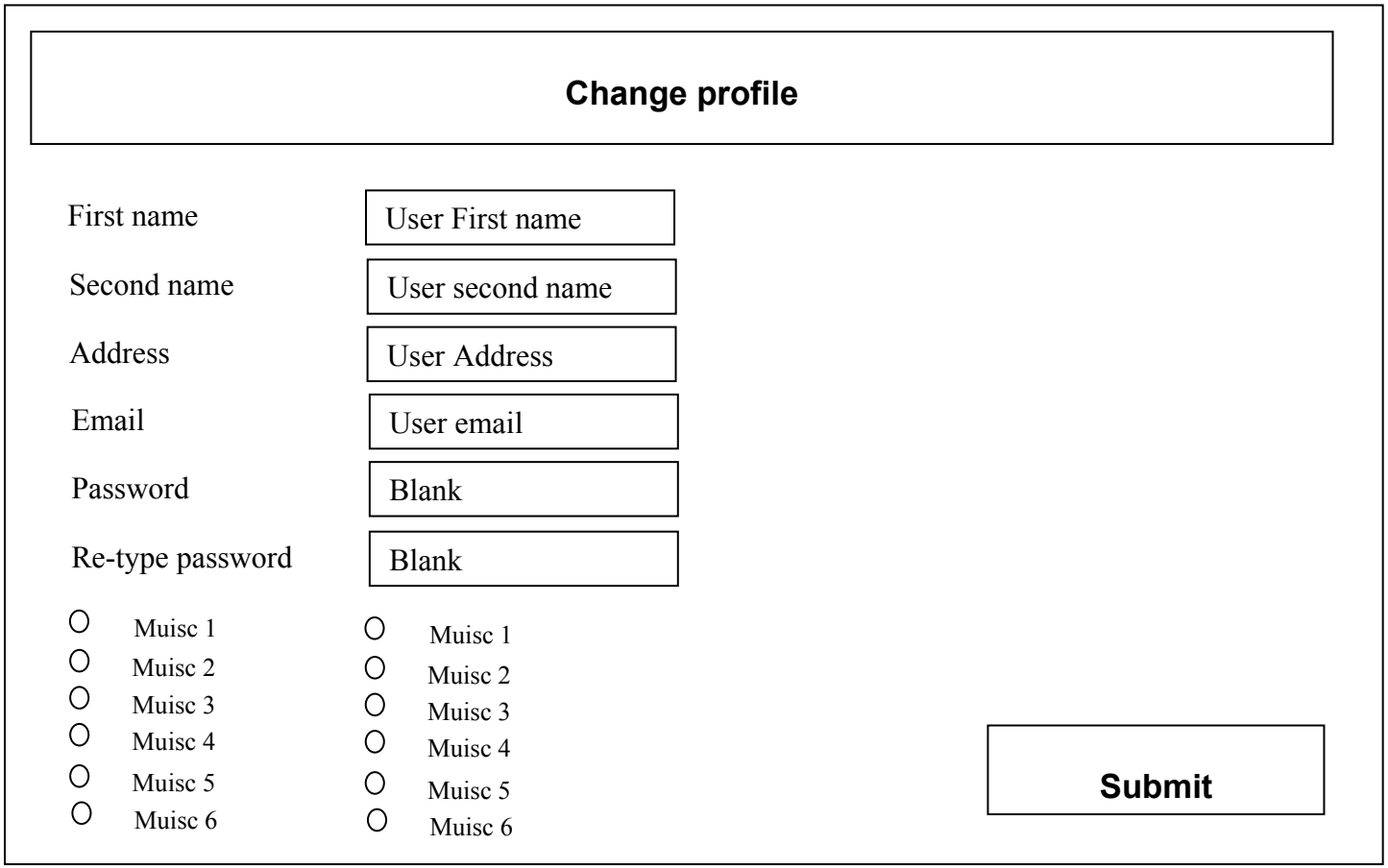
Existing user (via existing user link)



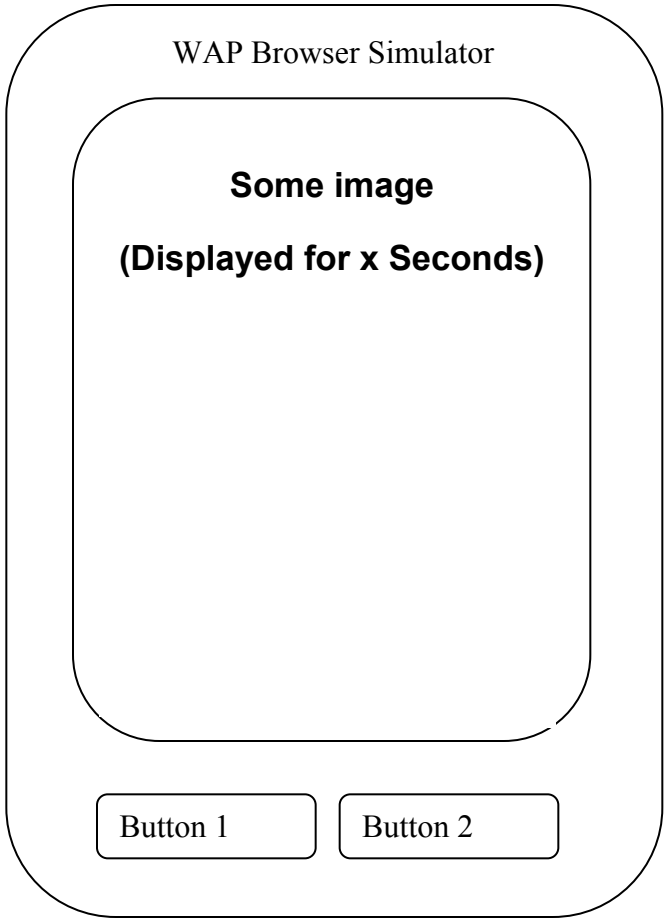
Personalised Page

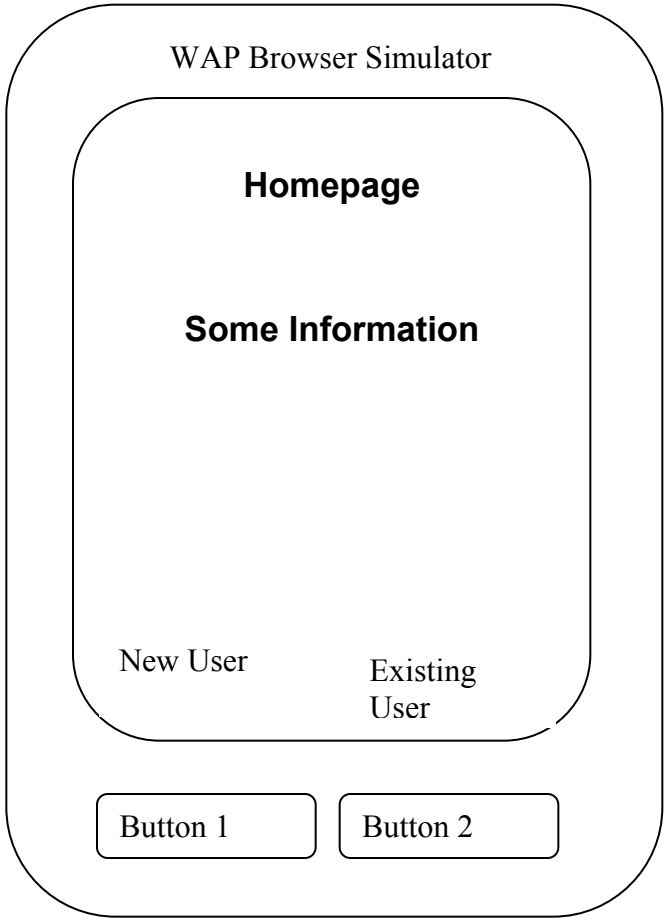


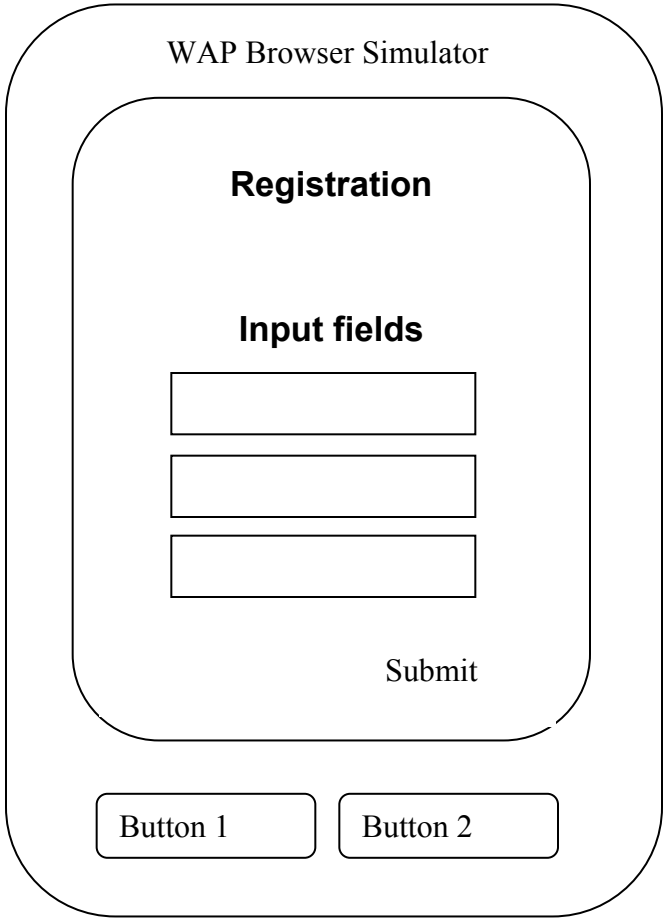
Change profile page – change profile link

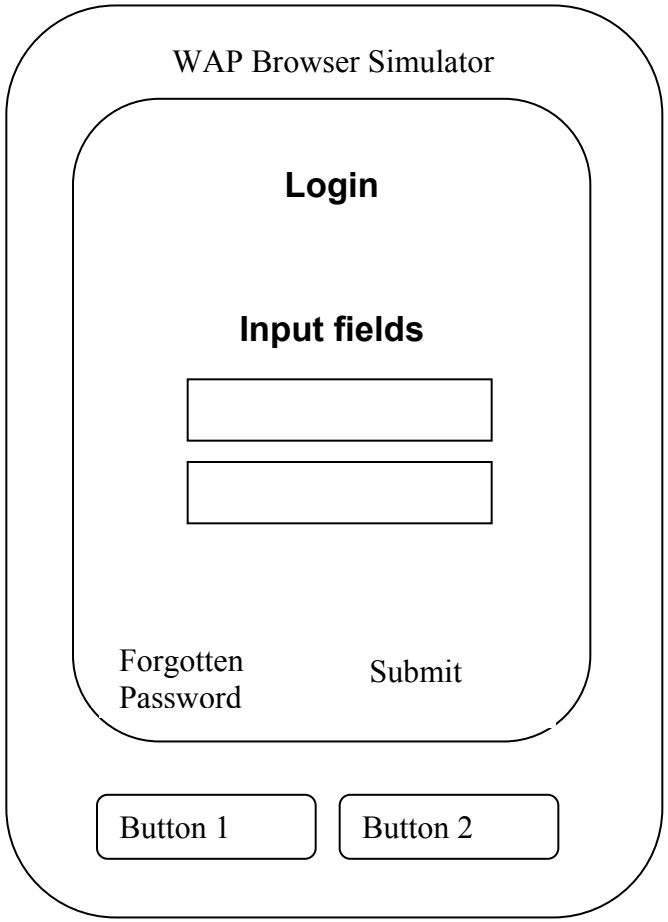


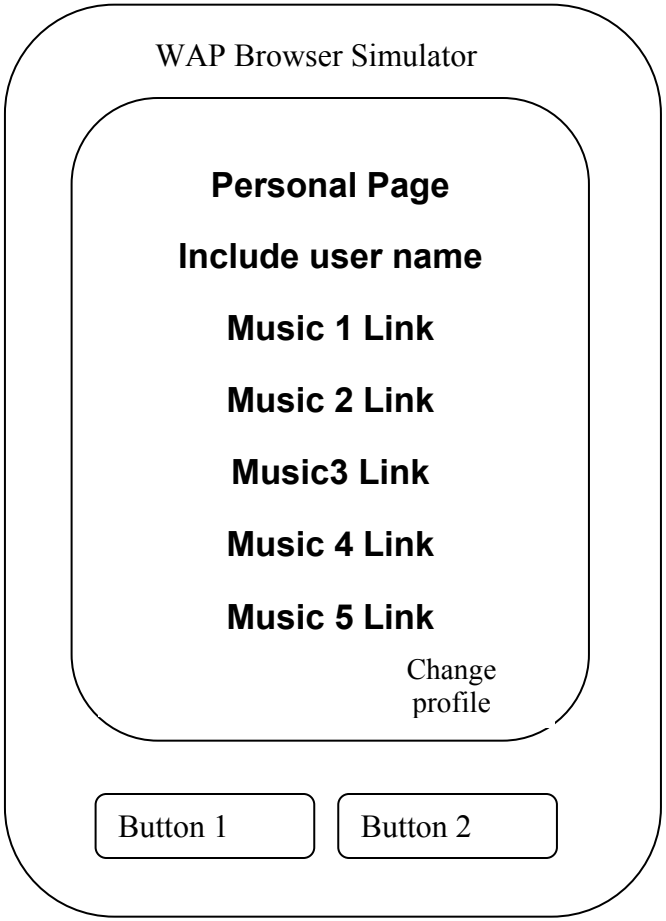
Wireless WML Browser Prototype –

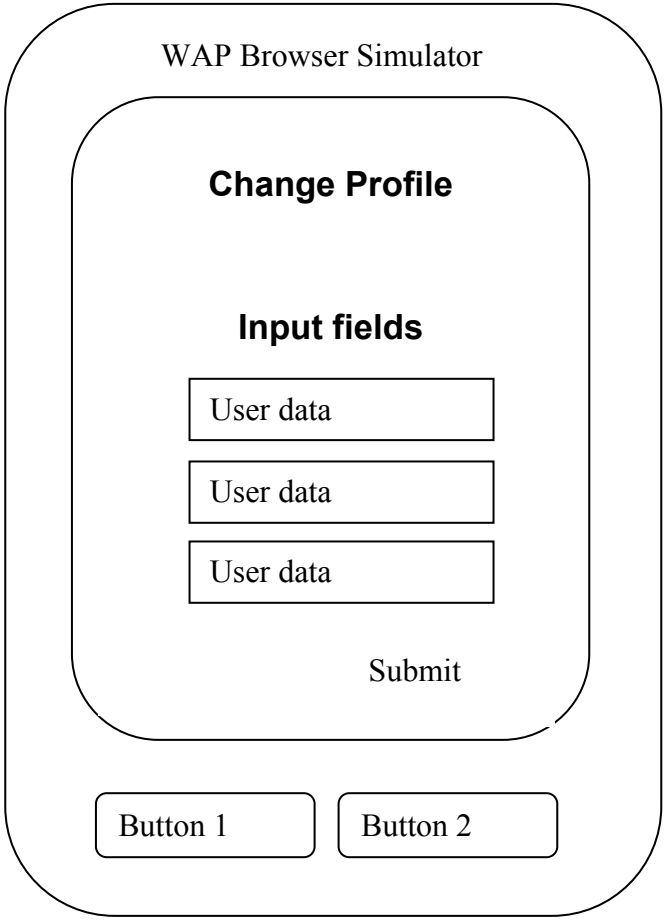





WAP Browser Simulator
Registration
Input fields
Submit
Button 1
Button 2


WAP Browser Simulator
Login
Input fields
Forgotten Password
Submit
Button 1
Button 2


WAP Browser Simulator
Personal Page
Include user name
Music 1 Link
Music 2 Link
Music3 Link
Music 4 Link
Music 5 Link
Change profile
Button 1
Button 2


WAP Browser Simulator
Change Profile
Input fields
User data
User data
User data
Submit
Button 1
Button 2

3.2 Class diagram

3.2.1 Noun identification

- System
- Wired Client
- Wireless Client
- Cookie
- Client Hard Disk
- Login Details
- Music Interest
- Music profile
- WAP device
- Web Browser
- WAP Browser
- Client
- Registration

- Wired Client and Wireless Client – These two nouns are users of the system although they cannot be classified as objects some data variables will be used to record the type of client attempting to access the system. For now I shall consider them as data for variables.

- Web Browser and WAP Browser – these two nouns have very close relationship to the different clients (above), as a WAP device has a WAP browser and Wired Client has a HTML browser. Therefore browser type will be considered as device type.

- WAP device also has a close relationship therefore this will also be abandoned.

Nouns to too vague for classes.

- Music Interest
- System – this comprises of all classes in the system
- Clients Hard disk – External to the system, no need to record as data or used in method.
- Client – Far to o vague, however some client data will need to be stored in customer details.

Classes found

- Registration – Some type of class will be needed to handle registration input

- Cookie –Although a cookie will not be a class, a class may be needed to deploy and read the cookie.

- Login Detail - The class itself wound be called login, which will verify password and user name.

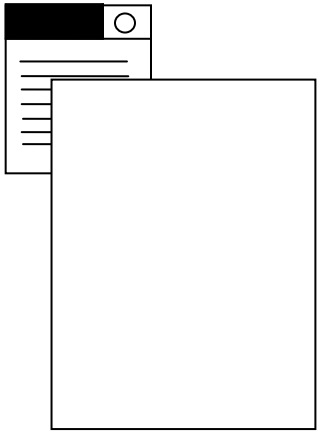
- Music Profile – Likely to be a bean of data created at runtime.

- Customer – A dynamically created bean holding customer details and other methods.

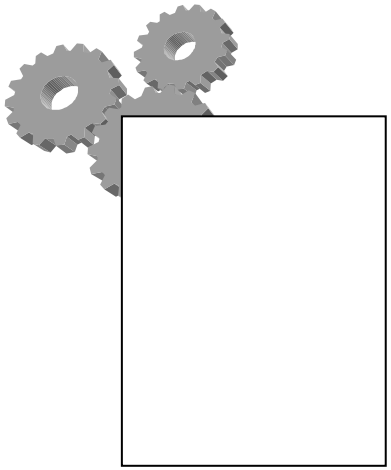
4. Design

Note: A clear distinction must be made before the documentation begins. Index.jsp is a JSP that redirects control to the main Servlet Controller; it is not a displayable page in itself, just a file that can be access by a WAP or PC device, the main index page which displays a welcome message is created on the fly using XML and XSL. Any reference to index page is aimed at the welcome page created on the fly.

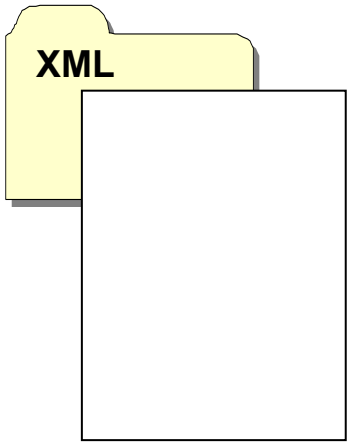
4.1.Web application extension



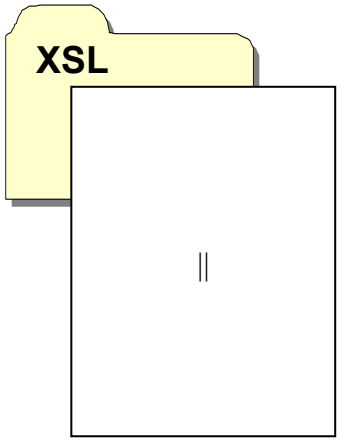
This icon represents a Java Server Page; a user can only communicate with a JSP or a mark-up language page i.e. HTML or WML, therefore this icon will act as the interface between the customer and the Servlets.



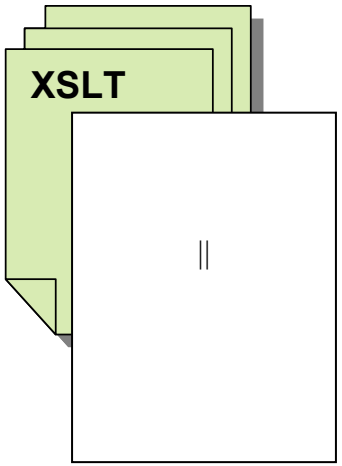
This icon represents a Java Servlet; this is used to process the data enter via a mark-up page or a JSP. Servlets run in the background therefore a user cannot have direct access to a Servlet.



An XML file is used as core content for a page; an xml file itself cannot be display and has no functionality to the user. XML file will not be allowed to interface with the customer. Instead it will be processed by a Servlet with its corresponding XSL file.



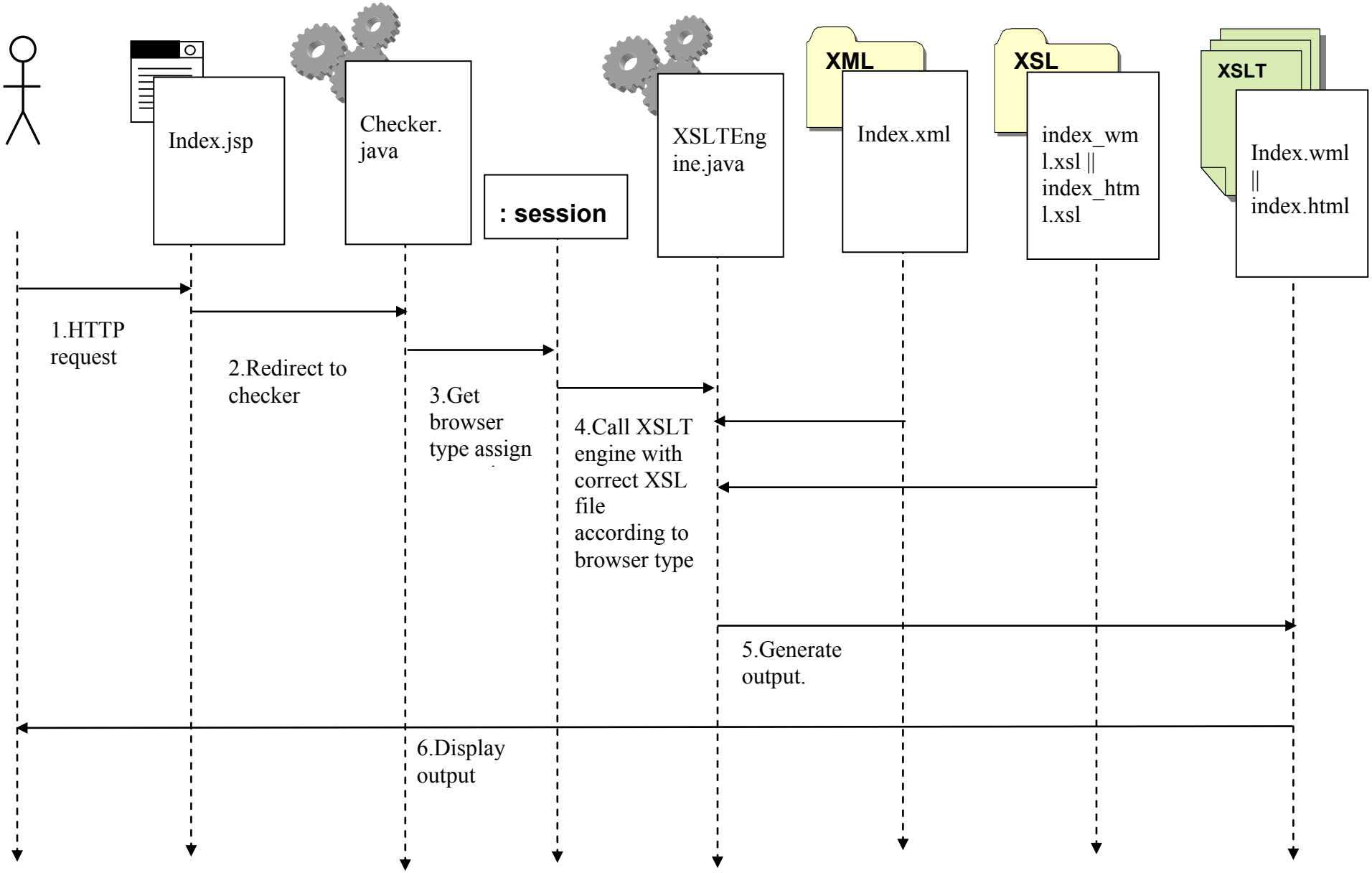
An XSL file is used to outline the transformation to be performed on the XML file; in itself it cannot be displayed to the user therefore it will not interface with the user. Like an XML file it will be processed by a Servlet with its corresponding XML file. The || (or) sign is used to indicate either a WML or HTML XSL file.



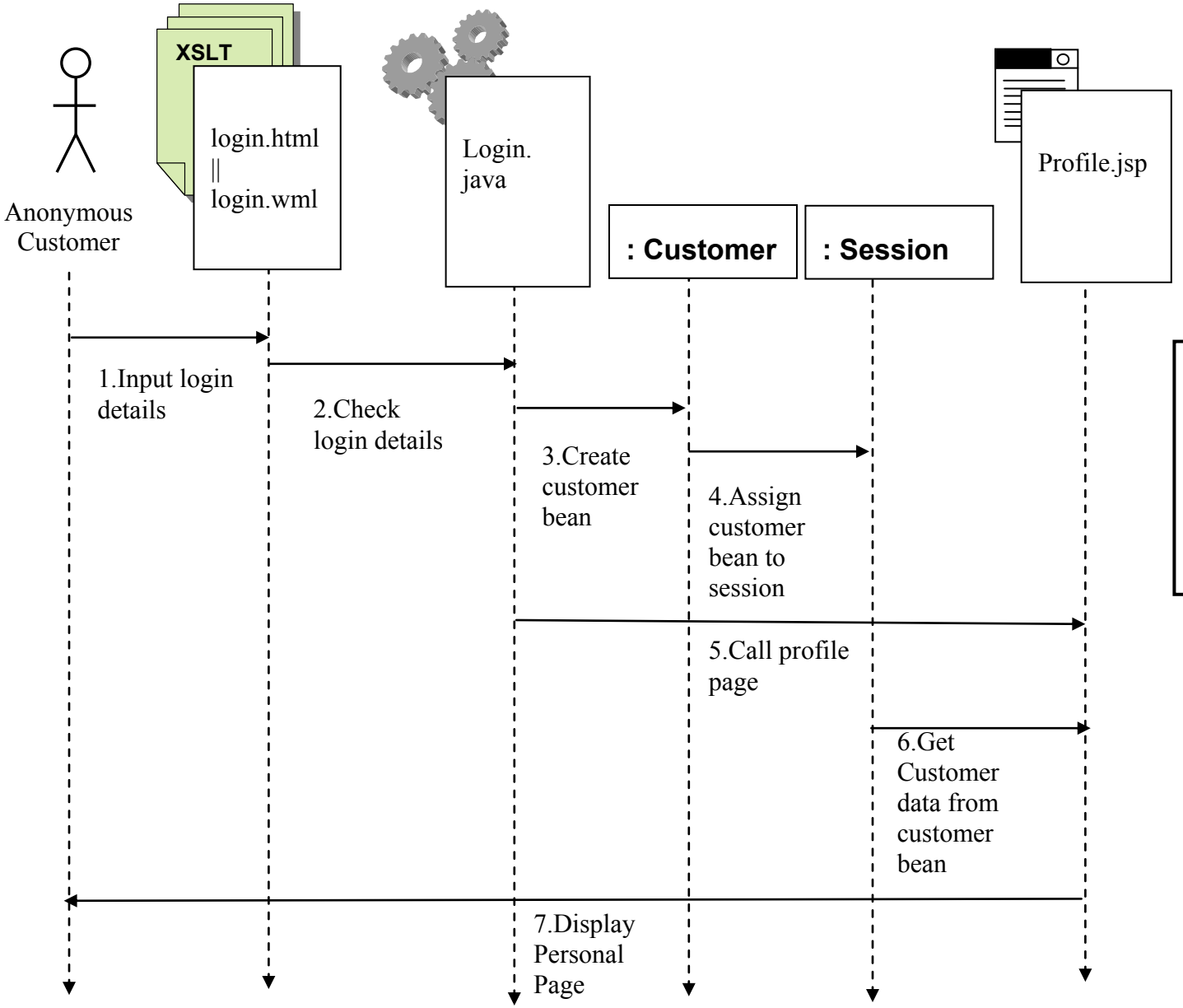
Although the icon reads XSLT this icon represents the output of the XML transformation, this could be WML or HTML. Therefore this icon along with a JSP will interface with the user and the Servlet. The || (or) sign is used to indicate either a WML or HTML file extension.

: bean or object

Main success scenario for UC1

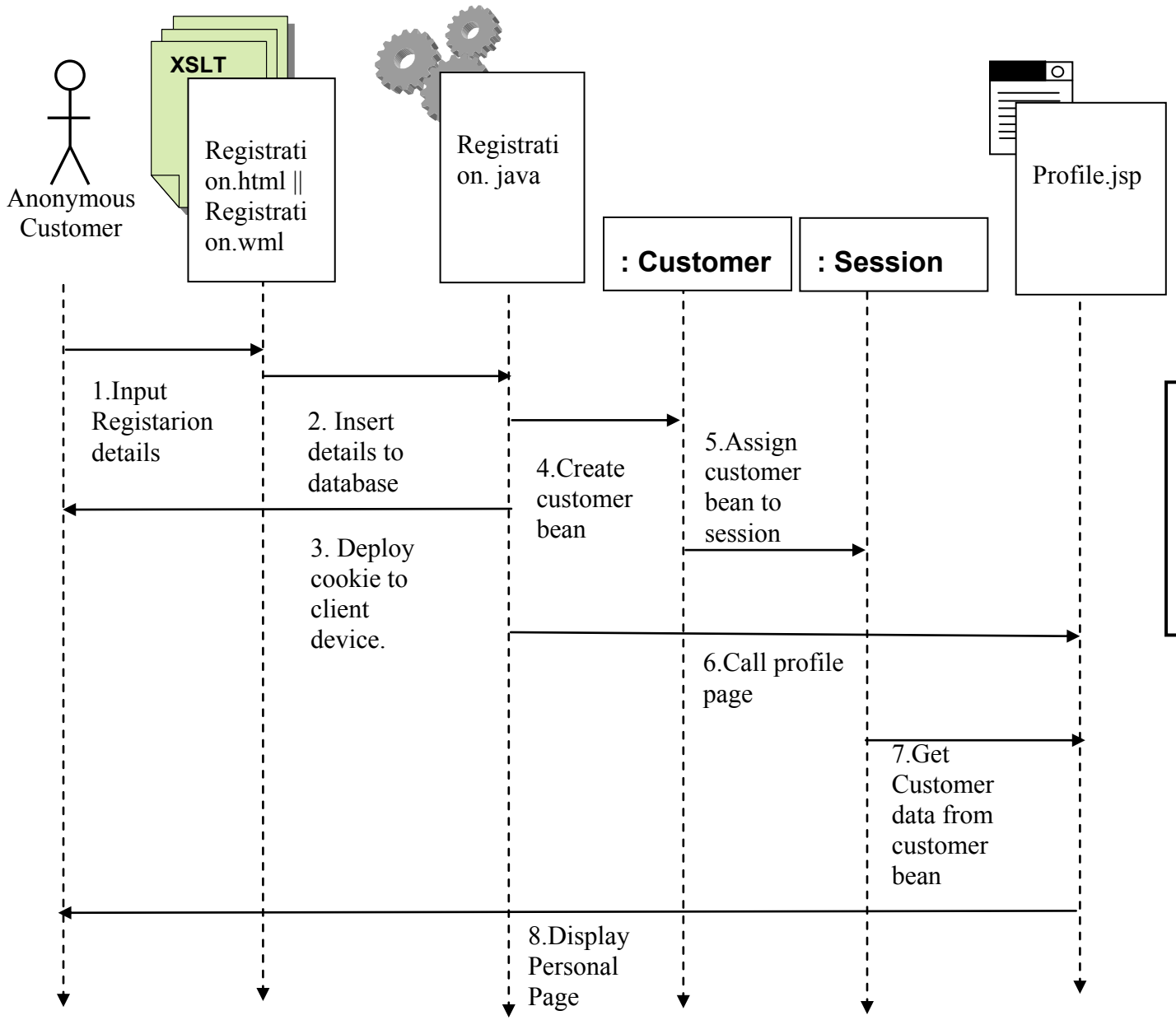


Main success scenario for UC2



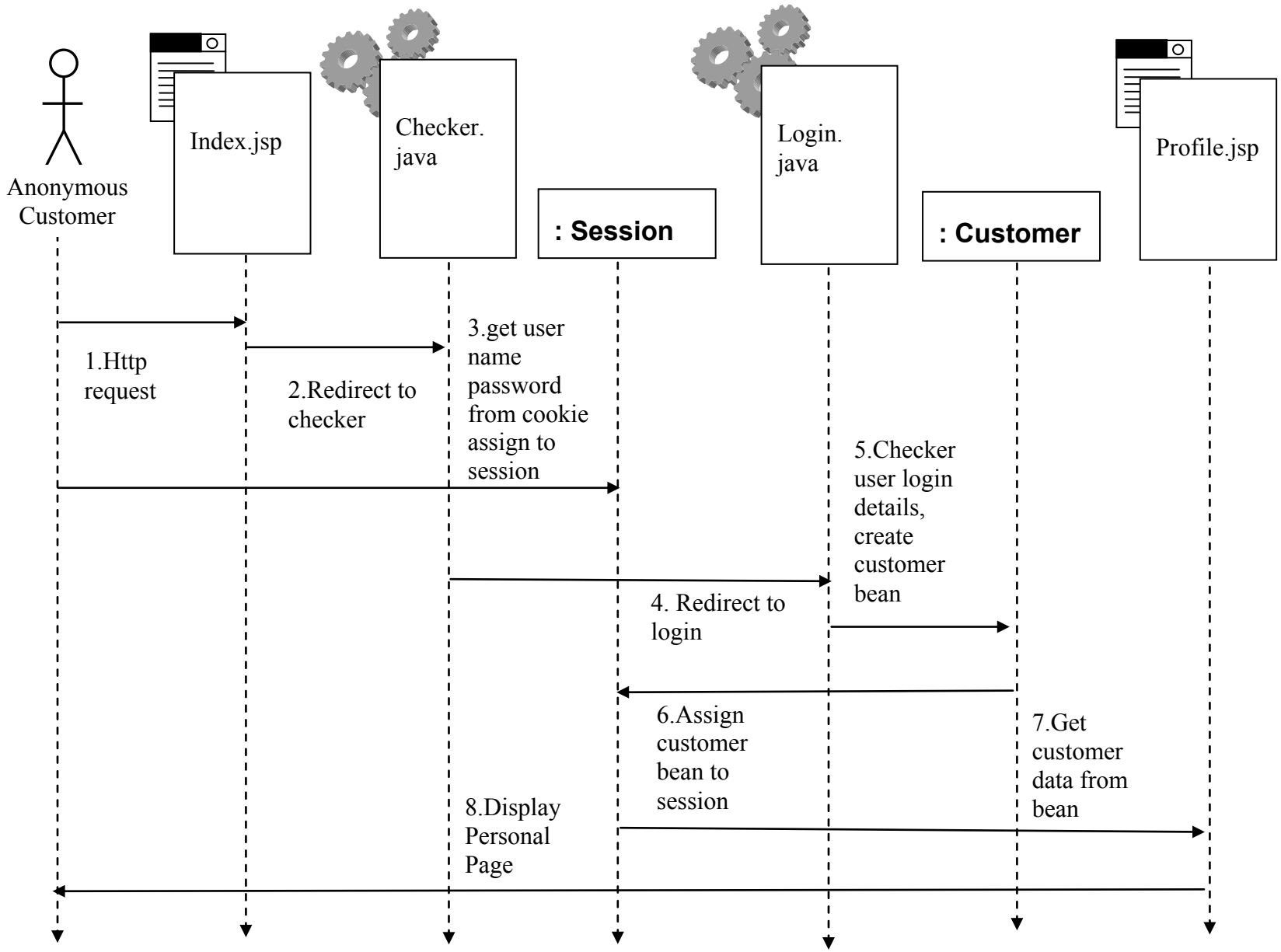
The first XSLT Page was create by calling the checker Servlet. A generic sequence diagram is show in Appendix

Main success scenario for UC3

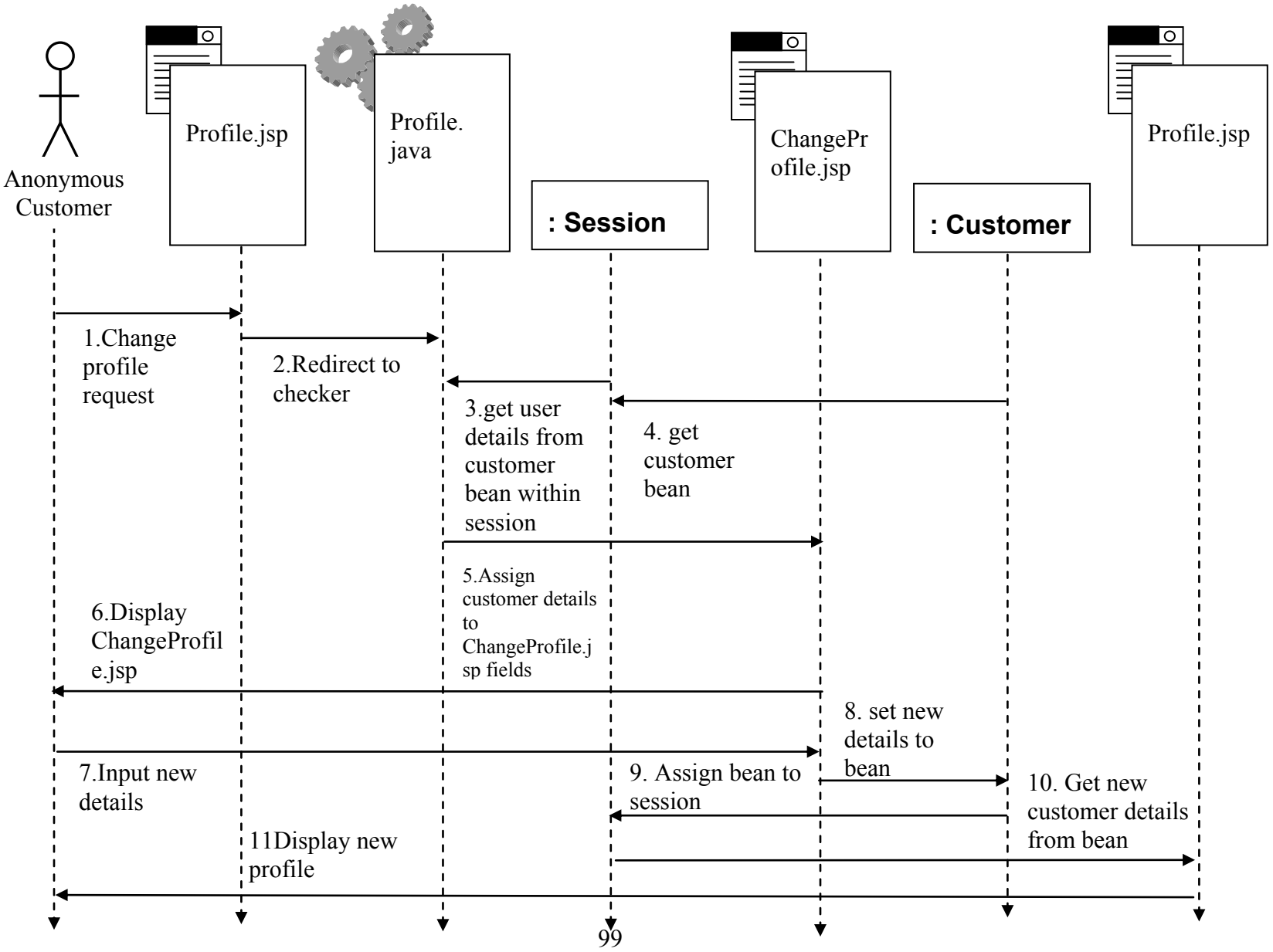


The first XSLT Page was create by calling the checker Servlet. A generic sequence diagram is show in Appendix

Main success scenario for UC4



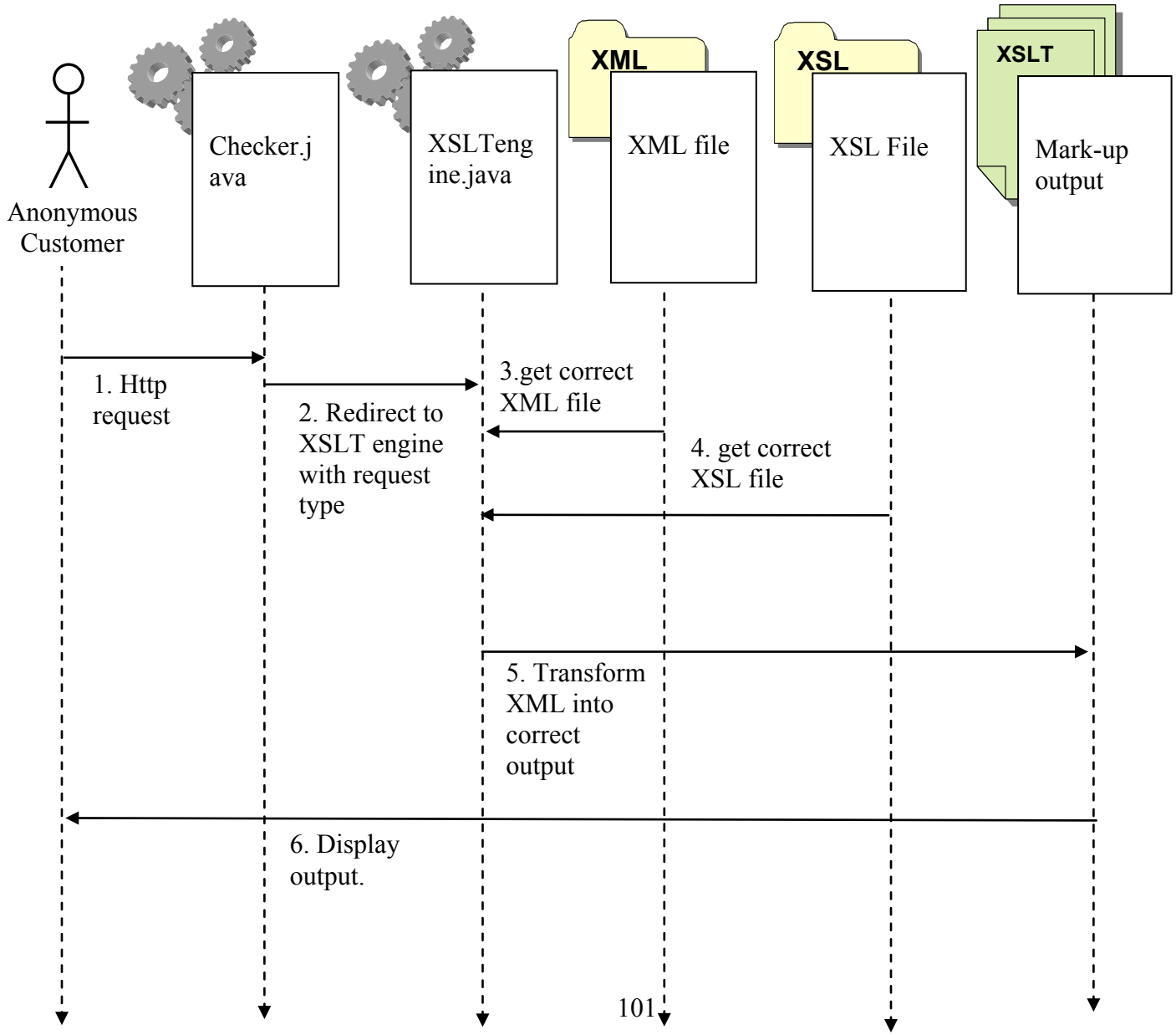
Main success scenario for UC5



Appendix F

Generic Sequence Diagram

Generic Http request

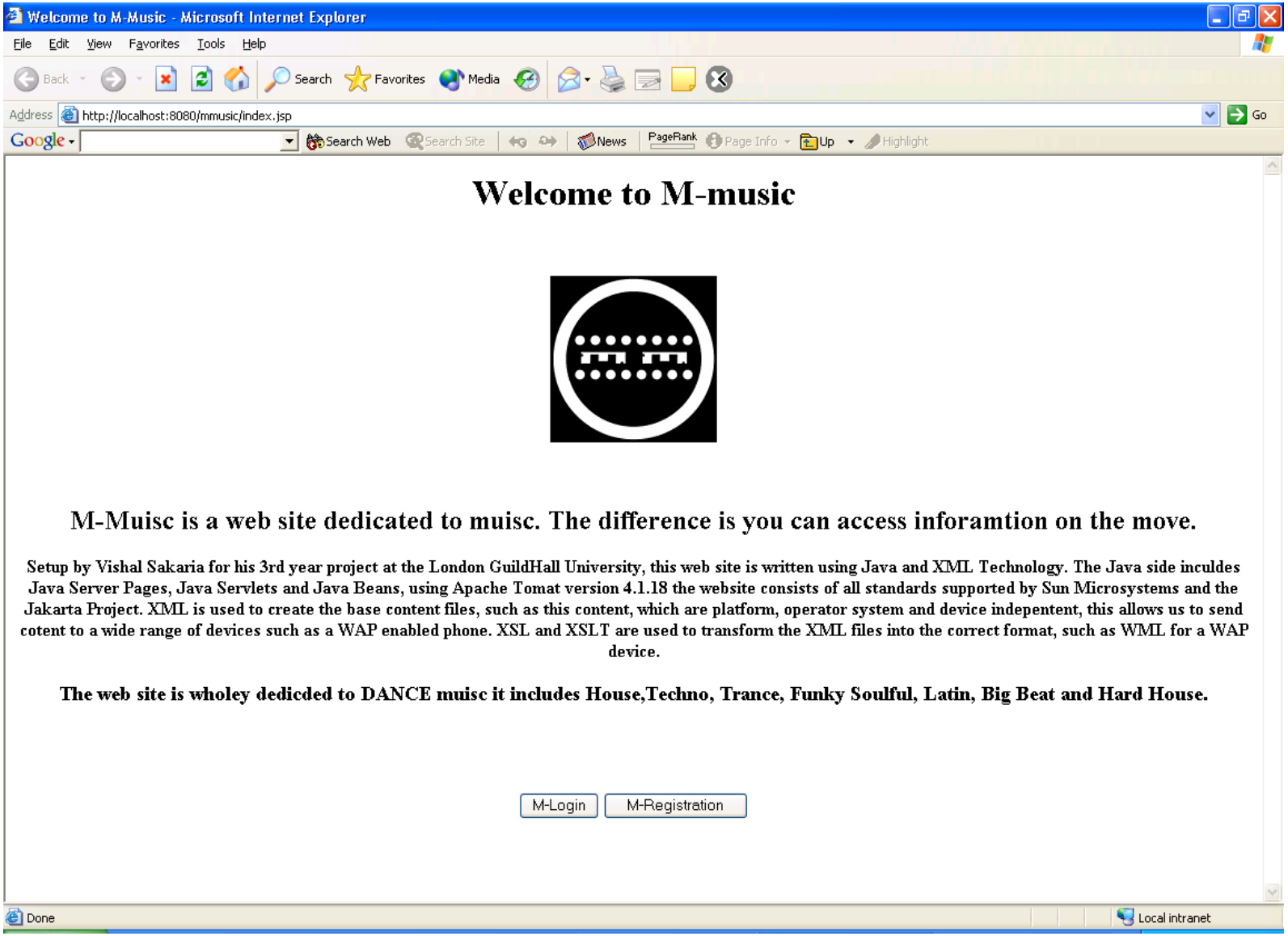


Appendix G

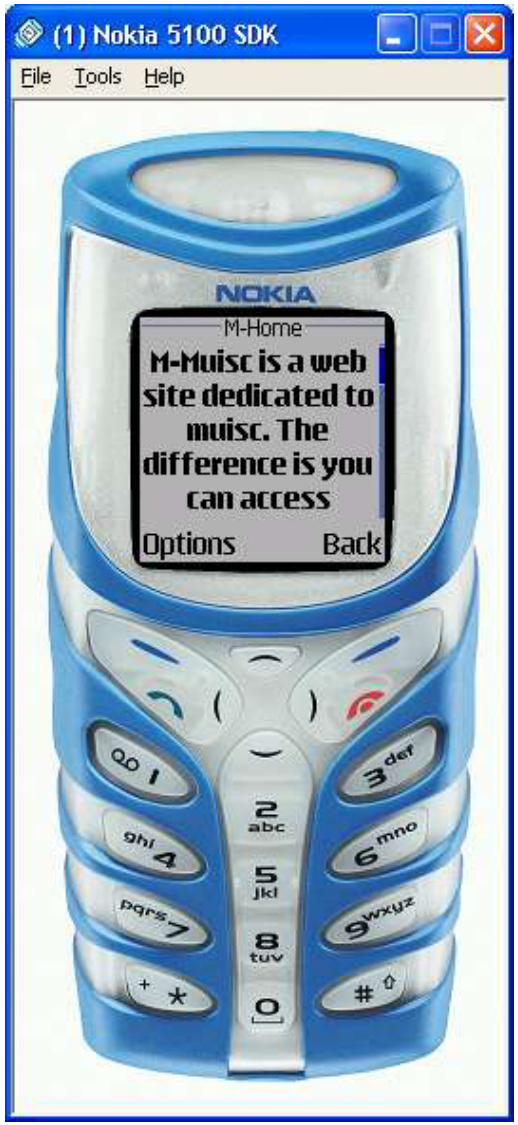
Testing

Test case for Use case 1 UC1

The Welcome page is displayed as I wish, with all content included.



The Welcome page is displayed as I wish, with relevant content included, the first screen dump show the m-music logo, this only appears for 4 seconds and then the text is shown.



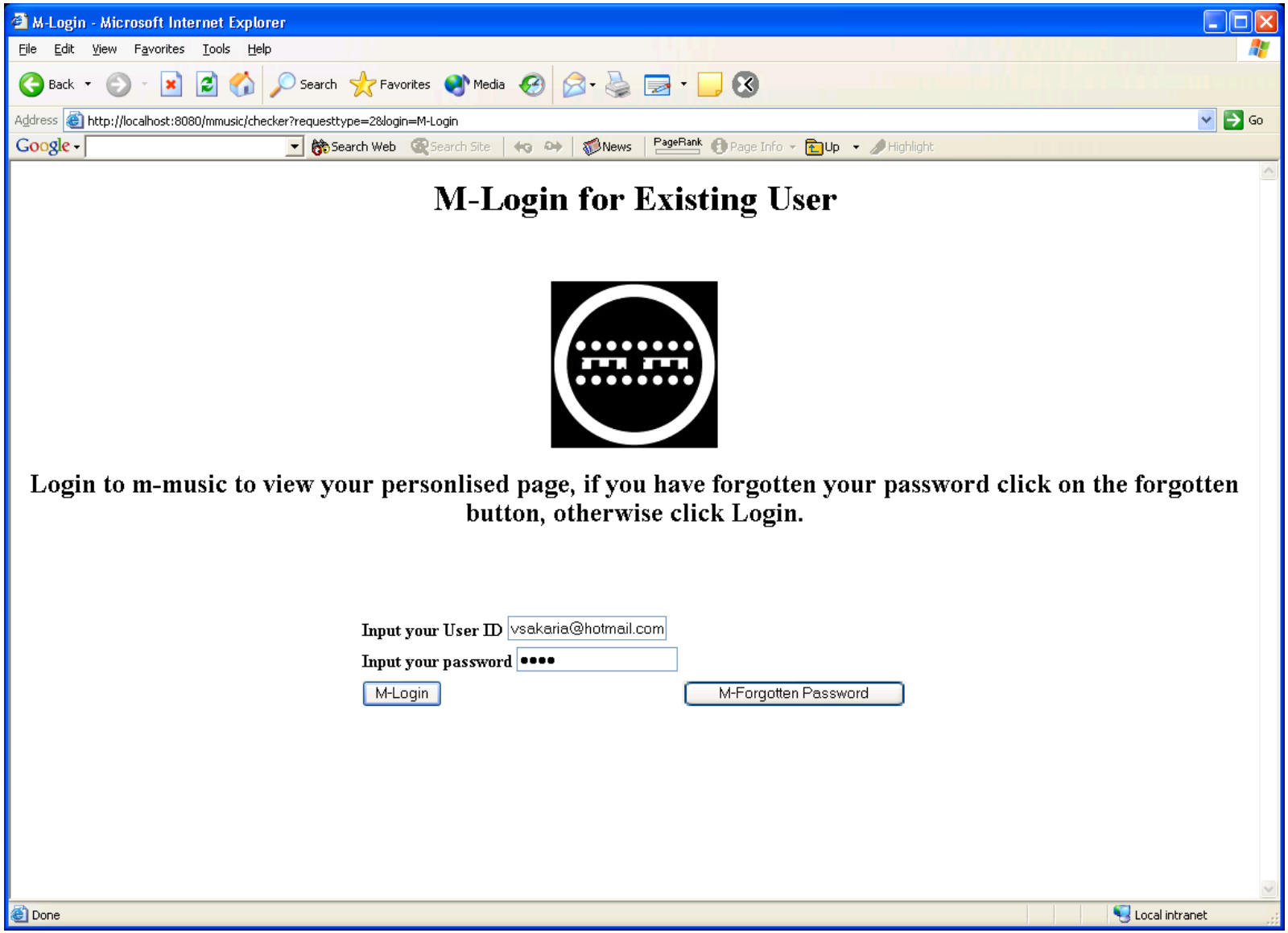

(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-Home
The web site is wholey dedicded to DANCE muisc it includes House,Techno, Trance, Funky Soulful,
Options Back

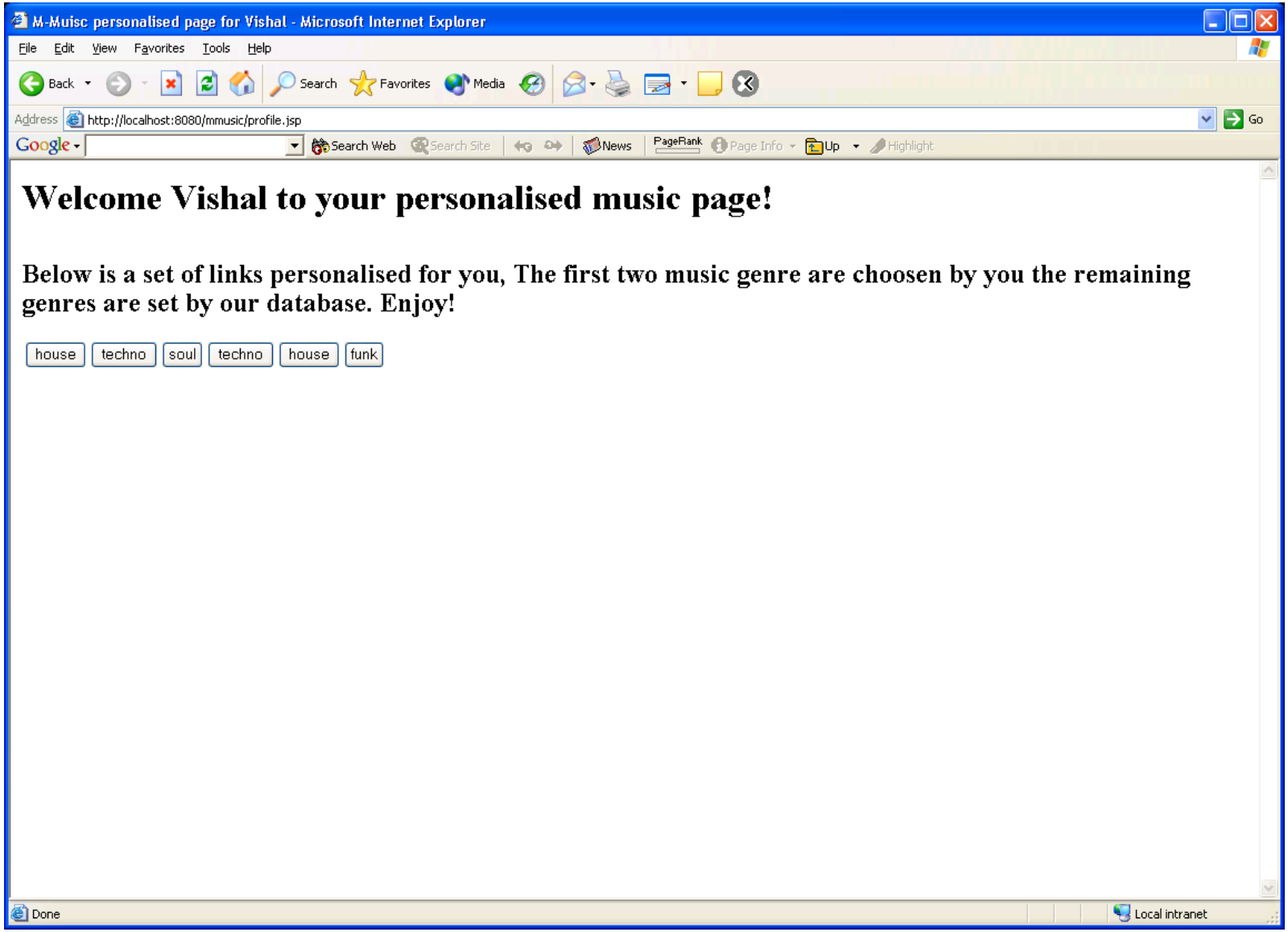

(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-Home
throught the site.
Enjoy!
Created by Vishal Sakaria using Java and XML
Options Back


(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-Home
Use the "option" button to navigate throught the site.
Enjoy!
Created by Vishal
Options Back

Test case for Use case 2 UC2

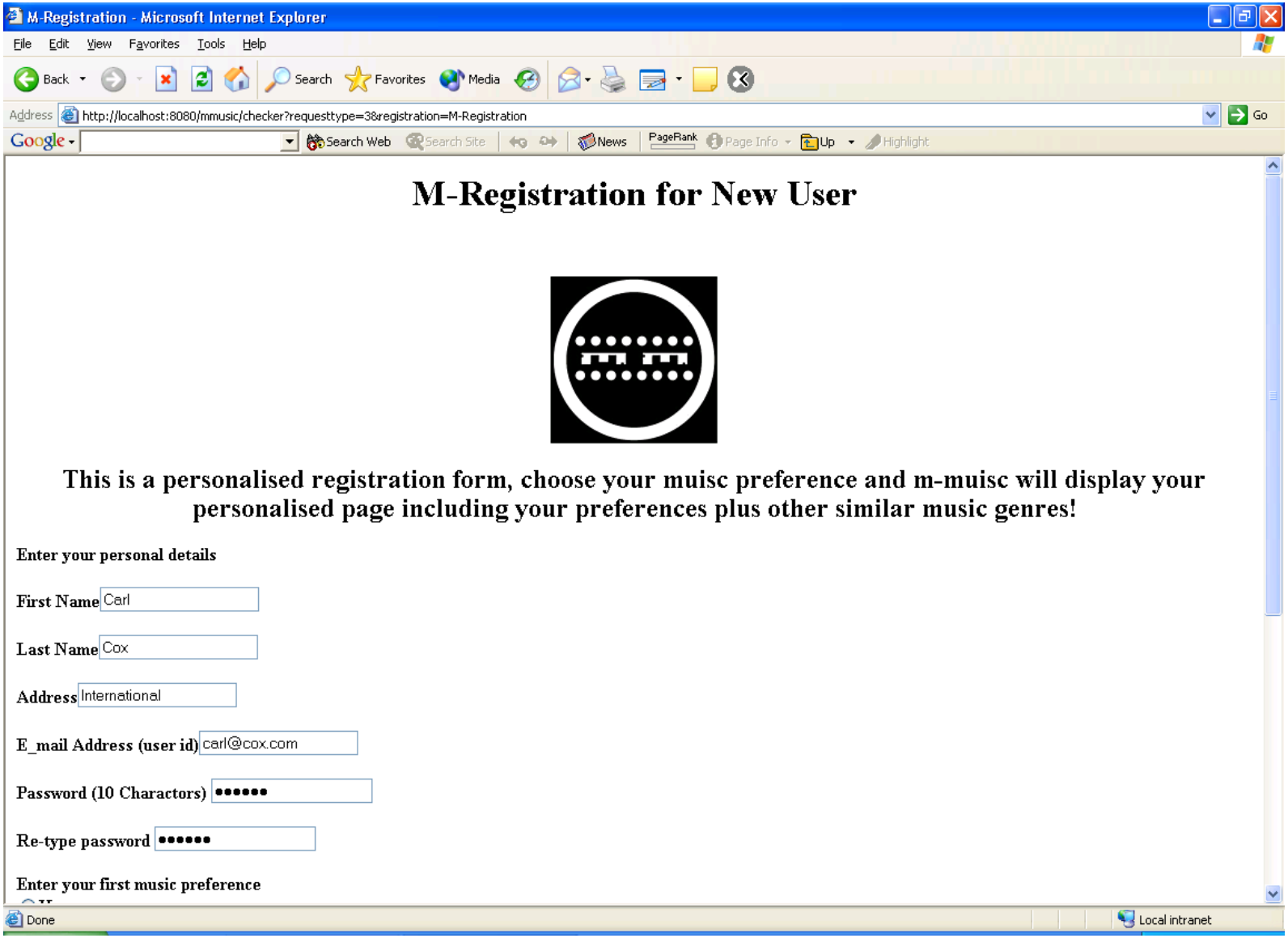
The result of this test are as I wished them to be, However some error existing in the Personalised page, music types are repeated this is down to the ruled base method of personalisation.

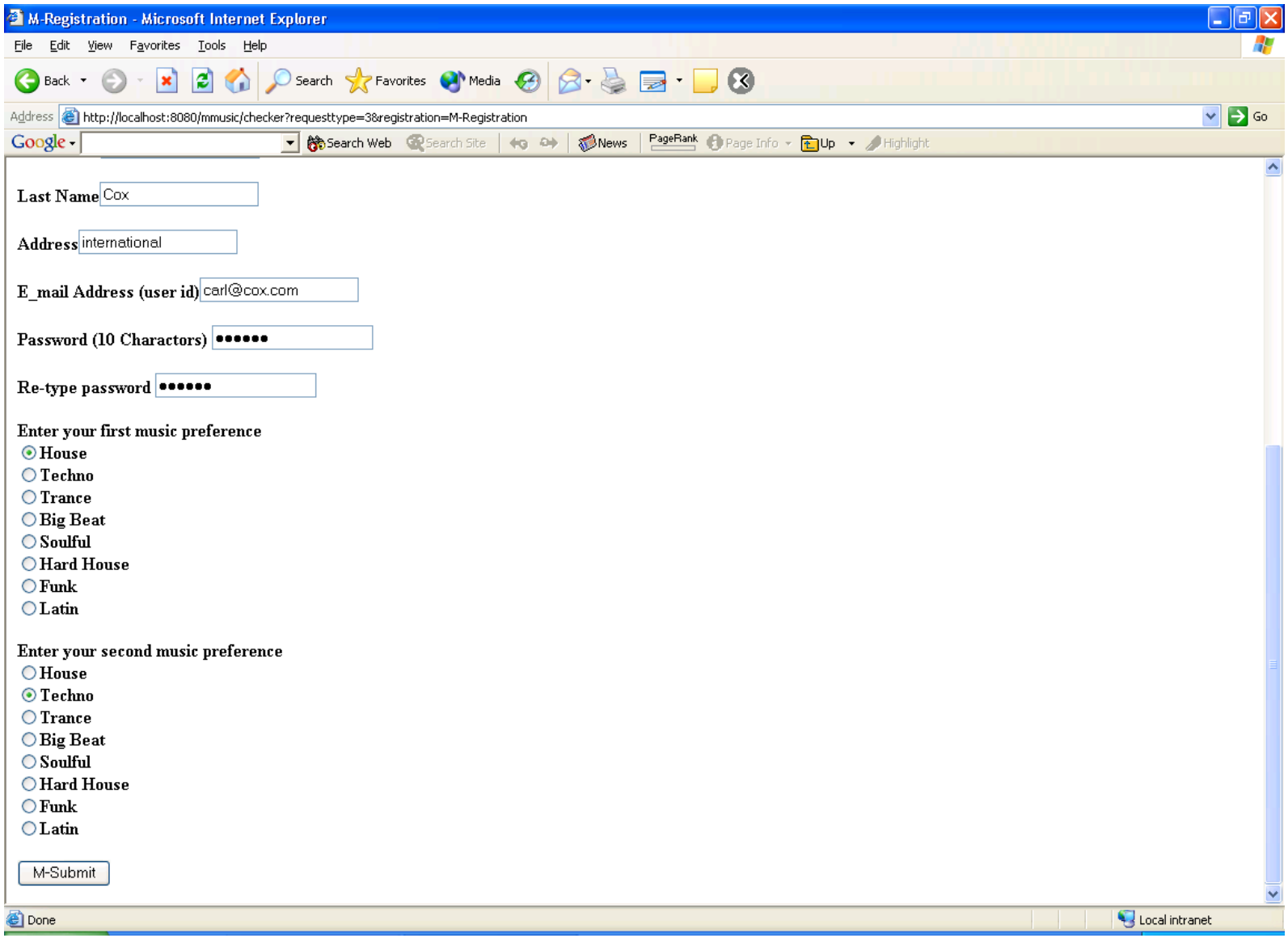


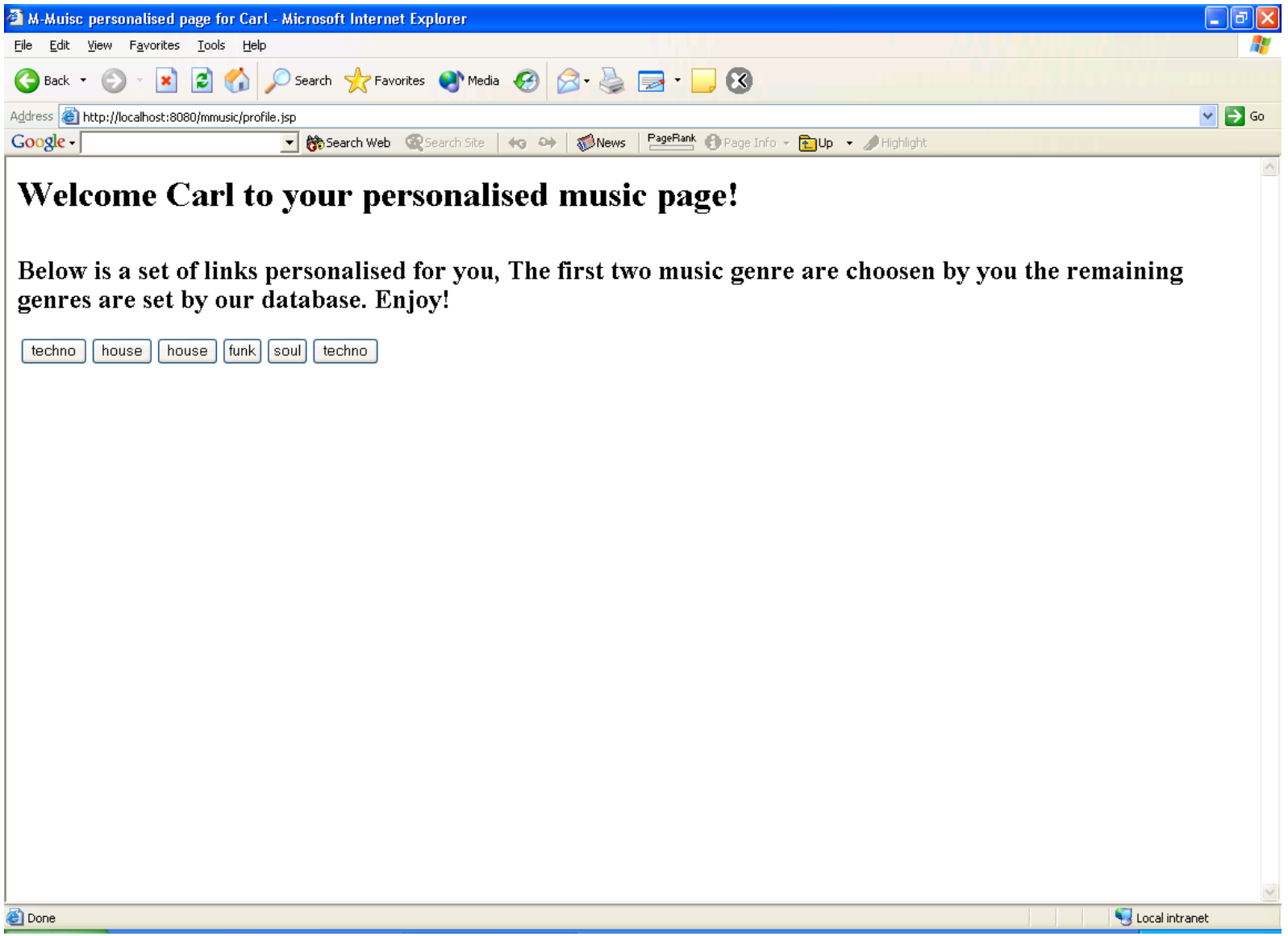

M-Muisc personalised page for Vishal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mmusic/profile.jsp
Go
Google Search Web Search Site News PageRank Page Info Up Highlight
Welcome Vishal to your personalised music page!
Below is a set of links personalised for you, The first two music genre are choosen by you the remaining genres are set by our database. Enjoy!
house techno soul techno house funk
Done
Local intranet

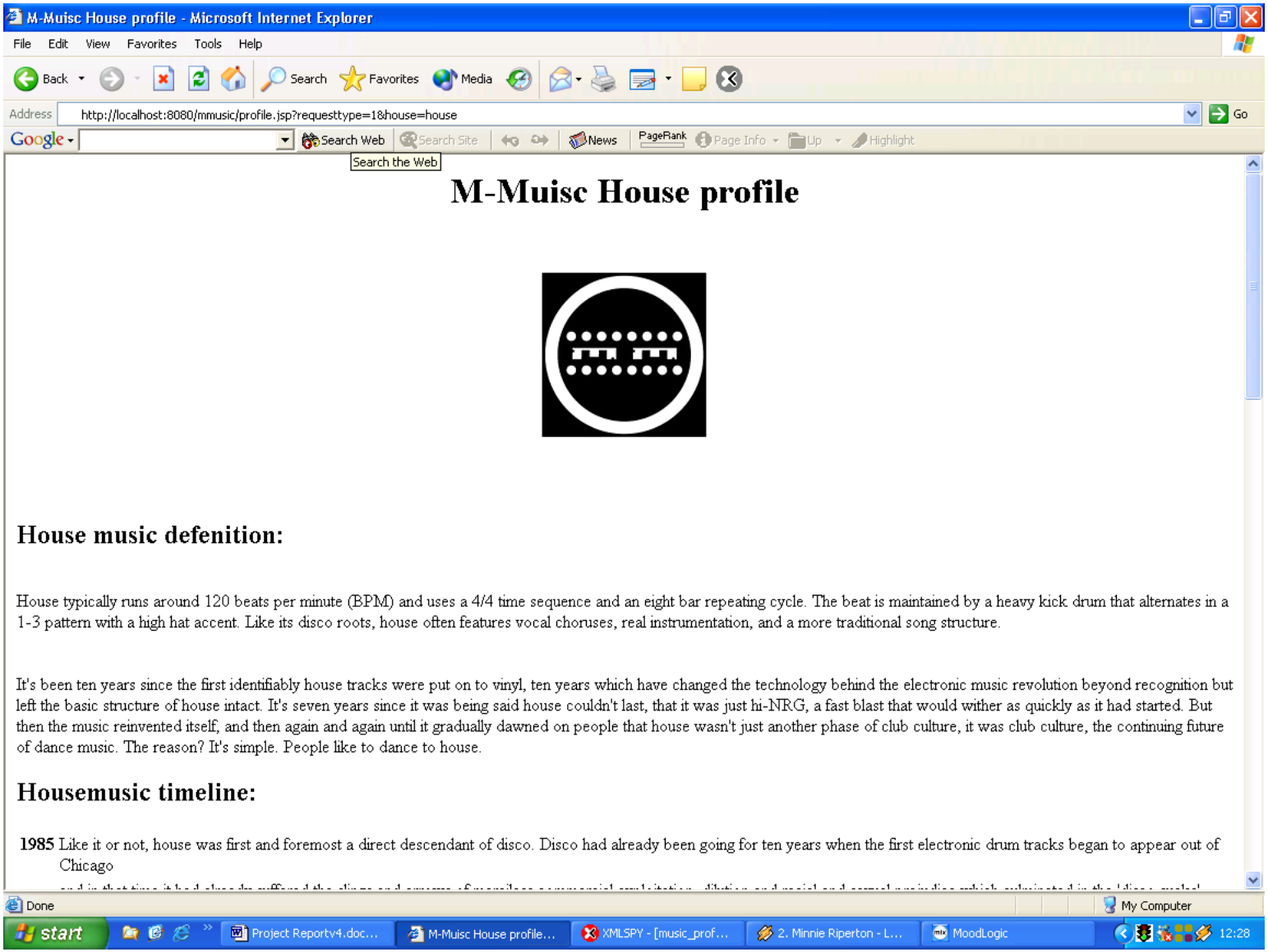
Test case for Use case 3 UC3

These results are as I wished again there is some error in the personalised page.




M-Registration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mmusic/checker?requesttype=3&registration=M-Registration
Go
Google Search Web Search Site News PageRank Page Info Up Highlight
Last Name Cox
Address international
E_mail Address (user id) carl@cox.com
Password (10 Charactors) ••••••
Re-type password ••••••
Enter your first music preference
House
Techno
Trance
Big Beat
Soulful
Hard House
Funk
Latin
Enter your second music preference
House
Techno
Trance
Big Beat
Soulful
Hard House
Funk
Latin
M-Submit
Done
Local intranet

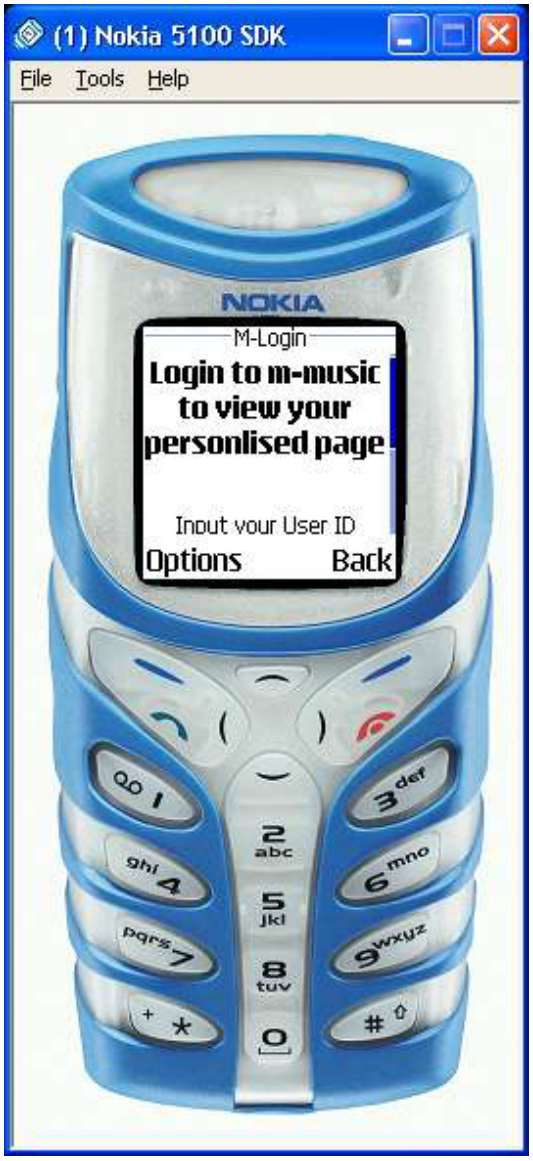

M-Muisc personalised page for Carl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mmusic/profile.jsp
Go
Google Search Web Search Site News PageRank Page Info Up Highlight
Welcome Carl to your personalised music page!
Below is a set of links personalised for you, The first two music genre are choosen by you the remaining genres are set by our database. Enjoy!
techno
house
house
funk
soul
techno
Done
Local intranet


M-Muisc House profile - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mmusic/profile.jsp?requesttype=1&house=house
Go
Google Search Web Search Site News PageRank Page Info Up Highlight
Search the Web
M-Muisc House profile
House music defenition:
House typically runs around 120 beats per minute (BPM) and uses a 4/4 time sequence and an eight bar repeating cycle. The beat is maintained by a heavy kick drum that alternates in a 1-3 pattern with a high hat accent. Like its disco roots, house often features vocal choruses, real instrumentation, and a more traditional song structure.
It's been ten years since the first identifiably house tracks were put on to vinyl, ten years which have changed the technology behind the electronic music revolution beyond recognition but left the basic structure of house intact. It's seven years since it was being said house couldn't last, that it was just hi-NRG, a fast blast that would wither as quickly as it had started. But then the music reinvented itself, and then again and again until it gradually dawned on people that house wasn't just another phase of club culture, it was club culture, the continuing future of dance music. The reason? It's simple. People like to dance to house.
Housemusic timeline:
1985 Like it or not, house was first and foremost a direct descendant of disco. Disco had already been going for ten years when the first electronic drum tracks began to appear out of Chicago
Done
My Computer
start
Project Reportv4.doc...
M-Muisc House profile...
XMLSPY - [music_prof...
2. Minnie Riperton - L...
MoodLogic
12:28

Modular Based Testing using XMLSpy built in XSLT engine

Login page,


(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-Login
Input your User ID
Input your password
Options
Back


(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-Login
Login to m-music to view your personlised page
Input your User ID
Options
Back

Registration Page

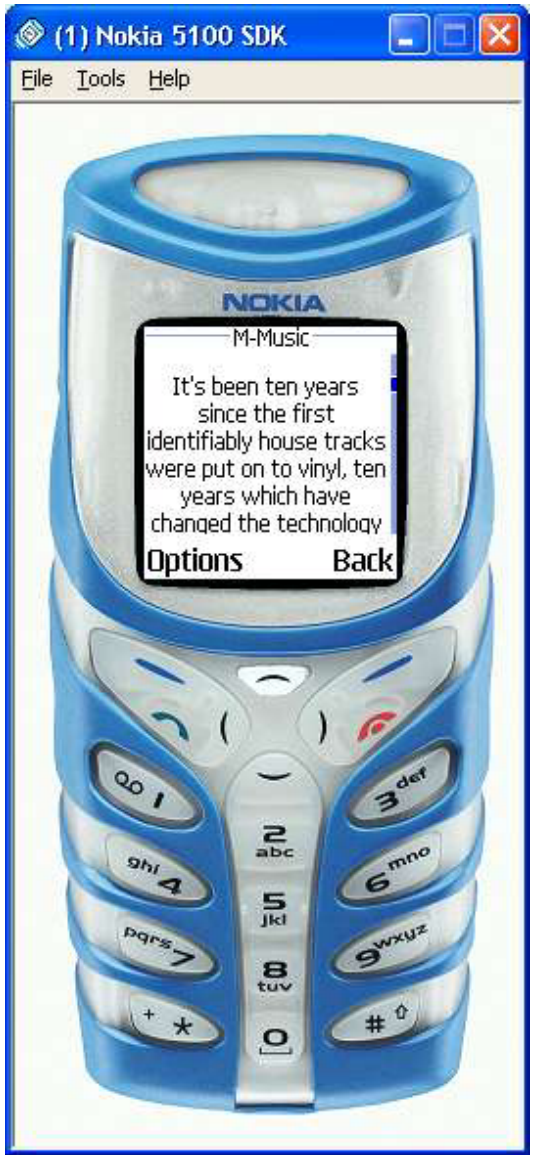

(1) Nokia 5100 SDK
File Tools Help
NOKIA
M-registration
Re-type password
Options
Back


(1) Nokia 5100 SDK
File Tools Help
NOKIA
Music
House
Techno
Trance
Big Beat
Mark
Done


(1) Nokia 5100 SDK
File Tools Help
NOKIA
Music
Soulful
Hard House
Funk
Latin
Mark
Done

Techno Music Profile

House Music Profile



Appendix G

Program Listing

Java File

Authenticator.java

```
package controller;

import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
//import controller.*;

public class Authenticator extends Detector
{
	String cook_login = "";
	String cook_pwd = "";


		boolean		cookieClient = cookieLocator( request, response );

		//Users has no session browser attribute and no cookie
		if( browserSession == null && cookieClient == false )
		{
			request.setAttribute( "target", Integer.toString( Page.INDEX ));
			session.setAttribute( "browser", Integer.toString( detectClient( request, response ) ) );
			dispatch.forward( request, response );
		}
		//User has no session browser attribute and cookie
		else if( browserSession == null && cookieClient == true )
		{
			request.setAttribute( "target", Integer.toString( Page.LOGIN ));
			session.setAttribute( "browser", Integer.toString( detectClient( request, response ) ) );
			session.setAttribute( "user_id", cook_login );
			session.setAttribute( "password_id", cook_pwd );
```

```
        dispatch.forward( request, response );

}

//User has a session browser attribute and no cookie
if( browserSession != null && cookieClient == false )
{
        String requestHandle = request.getParameter( "requesttype" );
        if( requestHandle != null )
        {
                request.setAttribute( "target", requestHandle );
        }
        else
        {
                request.setAttribute( "target", Integer.toString( Page.INDEX ));
        }
        dispatch.forward( request, response );
        //
}
//User has session browser attribute and cookie
else if( browserSession != null && cookieClient == true )
{
        String requestHandle = request.getParameter( "requesttype" );
        if( requestHandle != null )
        {
                request.setAttribute( "target", requestHandle );
        }
        else
        {
                request.setAttribute( "target", Integer.toString( Page.INDEX ));
        }
```

```
                dispatch.forward( request, response );
        }

        //dispatch.forward( request, response );
}

public boolean cookieLocator( HttpServletRequest request, HttpServletResponse response )
{
        Cookie mmusiccookies[] = request.getCookies();

        boolean cook_login_ok = false;
        boolean cook_pwd_ok = false;

        if( mmusiccookies != null )
        {
        for( int i=0; i < mmusiccookies.length; i++ )
        {
                String cookieName = mmusiccookies[i].getName();

                if( cookieName.equalsIgnoreCase("mmuser") )
                {
                        cook_login = mmusiccookies[i].getValue();
                        cook_login_ok = true;

                }
                if( cookieName.equalsIgnoreCase("mmpwd") )
                {
                        cook_pwd = mmusiccookies[i].getValue();
                        cook_pwd_ok = true;
                }
        }
```

```
                }

                if( cook_login_ok && cook_pwd_ok )
                return true;
                else
                return false;
        }
}
```

```
        /*
                Try retrieving the predefined cookie variables (eg. mmuser,mmpwd)

                check if it is null
                if null, redirect the user to login
                else, redirect the user to music page

                -----> Storing a cookie in the client's machine after successful login

                Cookie mmuser = new Cookie("mmuser",login);
                Cookie mmpwd = new Cookie("mmpwd",password);

                response.addCookie(mmuser);
                response.addCookie(mmpwd);
```

```
----> retrieval

Cookies mmusiccookies[] = request.getCookies();
String cook_login = "";
String cook_pwd = "";
boolean cook_login_ok = false;
boolean cook_pwd_ok = false;

for (int i=0;i<mmusiccookies.length;i++)
{
        String name = mmusiccookies[i].getName();

        if(name.equalsIgnoreCase("mmuser"))
        {
                cook_login = mmusiccookies[i].getValue();
                cook_login_ok = true;

        }
        if(name.equalsIgnoreCase("mmpwd"))
        {
                cook_pwd = mmusiccookies[i].getValue();
                cook_pwd_ok = true;
        }

        if(cook_login_ok && cook_pwd_ok)
                break;
}


*/
```

```
        //I want to set his browser settings here and store it in the session, therefore it will only need to be
                        // checked once and other servelts will simply reference session Browser variable via.
                        //I want to call AuthContoller servlet to do some work and then give control back to this
class
                        //I think this class should extend Action class

                        //dispatch.forward( request, response );
```

Detector

```
package controller;

import javax.servlet.*;
import javax.servlet.http.*;

import java.io.IOException;
import java.util.Enumeration;



/**
*	This class is the superclass for all controllers which will the
*	contact point for all client requests.
*	Function common to all controllers will be defined in this class
*
*/

public class Detector extends HttpServlet
{
	public final static int PC = 1;
	public final static int WAP = 2;

	private String browserName = "";
	private String mimeTypes = "";

	ServletContext sctx;

	public void init(ServletConfig sc) throws ServletException
	{
		sctx = sc.getServletContext();
	}
```

```
/**
*	@return Returns the ServletContext of that Servlet Engine
*
*/

	public ServletContext getServletContext()
	{
		return sctx;
	}

/**
*	This class detects the type of client whether wap client or html client
*	@param request - takes a javax.servlet.http.HttpServletRequest and
*	@param response - takes a javax.servlet.http.HttpServletResponse as arguments
*/

	protected int detectClient(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
	{
			int clientDetected = 0;

			browserName = request.getHeader("User-Agent");
			mimeTypes     = request.getHeader("Accept");

			browserName = browserName.toLowerCase();
			mimeTypes = mimeTypes.toLowerCase();

			/*
				If the browser is Mozilla compatible, then it
```

```
                        is PC only
                */

                if (browserName.indexOf("mozilla")!=-1)
                {
                        response.setContentType("text/html");
                        clientDetected = Device.PC;
                }


                if (mimeTypes.indexOf("text/vnd.wap.wml")!=-1)
                {
                        response.setContentType("text/vnd.wap.wml");
                        clientDetected = Device.WAP;
                }

                /*
                if (browserName.indexOf("j2me")!=-1)
                {
                        clientDetected = J2ME;
                }
                */

                return clientDetected;
        }

/**
*       Returns the Name of the client's Browser
*
*/
```

```
	protected String getBrowserName()
	{
		return browserName;
	}

/**
*	Returns the Mime Types supported by that particular client
*	browser. The Mime Types are separated by a delimiter ',' (comma)
*	It can be separated using @see java.util.StringTokenizer
*/

	protected String getMimeTypes()
	{
		return mimeTypes;
	}


}
```

Location.java

```
package controller;

import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Location extends HttpServlet
{
	public void doGet( HttpServletRequest request, HttpServletResponse response ) throws
ServletException, IOException
	{

		HttpSession session = request.getSession();

		//Get target page variable from request convert to int.
		String targetPageString = ( String ) request.getAttribute( "target" );

		int targetPage = Integer.parseInt( targetPageString );

		//Get browser type from session convert to int
		String browserSessionString = (String) session.getAttribute( "browser" );
		int browserSession = Integer.parseInt( browserSessionString );

		RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/xslt");

		if( targetPage == Page.INDEX )
		{
			if( browserSession == Device.PC )
			{
				request.setAttribute( "xmlfile", "index.xml" );
```

```
                request.setAttribute( "stylesheet", "index_HTML.xsl" );
                System.out.println( "PC client detected " );
                dispatcher.forward( request, response );

                //response.sendRedirect( "/mmusic/testH.html" );

        }
        else if( browserSession == Device.WAP )
        {
                response.setContentType("text/vnd.wap.wml");
                //response.sendRedirect( "/mmusic/testW.html" );

                request.setAttribute( "xmlfile", "index.xml" );
                request.setAttribute( "stylesheet", "index_WML.xsl" );
                dispatcher.forward( request, response );
        }
}

if( targetPage == Page.LOGIN )
{
        if( browserSession == Device.PC )
        {
                request.setAttribute( "xmlfile", "login.xml" );
                request.setAttribute( "stylesheet", "login_HTML.xsl" );
                dispatcher.forward( request, response );
        }
        else if( browserSession == Device.WAP )
        {
                System.out.println("WAP device detected");
                request.setAttribute( "xmlfile", "login.xml" );
                request.setAttribute( "stylesheet", "login_WML.xsl" );
```

```
                        dispatcher.forward( request, response );
                    }
                }

                if( targetPage == Page.REGISTRATION )
                {
                    if( browserSession == Device.PC )
                    {
                        request.setAttribute( "xmlfile", "registration.xml" );
                        request.setAttribute( "stylesheet", "registration_HTML.xsl" );
                        dispatcher.forward( request, response );
                    }
                    else if( browserSession == Device.WAP )
                    {
                        request.setAttribute( "xmlfile", "registration.xml" );
                        request.setAttribute( "stylesheet", "registration_WML.xsl" );
                        dispatcher.forward( request, response );
                    }
                }

                if( targetPage == Page.MUSIC )
                {
                    RequestDispatcher dispatcherChecker =
getServletContext().getRequestDispatcher("/checker");

                    if( browserSession == Device.PC )
                    {
                        response.sendRedirect( "/mmusic/profile.jsp" );
                        //request.setAttribute( "xmlfile", "registration.xml" );
                        //request.setAttribute( "stylesheet", "registration_HTML.xsl" );
                        //dispatcher.forward( request, response );
```

```
                        //dispatcherChecker.forward( request, response );
                        //need to go to profile page
                }
                else if( browserSession == Device.WAP )
                {
                        dispatcherChecker.forward( request, response );
                }
        }


    }
}
```

Page.java

```
package controller;

public class Page
{
	public static final int INDEX = 1;
	public static final int LOGIN = 2;
	public static final int REGISTRATION = 3;
	public static final int MUSIC = 4;
}
```

Device.java

```
package controller;

public class Device
{
	public static final int PC = 1;
	public static final int WAP = 2;
}
```

Customer.java

```
package handler;

import java.sql.*;

public class Customer
{
	Connection connection;

	//Database Driver variables

	String serverName = "localhost";
       String mydatabase = "music";
       String url = "jdbc:mysql://" + serverName +  "/" + mydatabase; // a JDBC url
       String username = "admin";
       String password = "jdbc";
	String driverName = "org.gjt.mm.mysql.Driver"; // MySQL MM JDBC driver


	//String db_password;


	//Bean Variables
	private int customer_id;
	private String customer_login;
	private String first_name;
	private String last_name;
	private String address;
	private String customer_password;
	private String customer_music_1;
	private String customer_music_2;
	private String music_1_1;
```

```
	private String music_1_2;
	private String music_2_1;
	private String music_2_2;

	public Customer( String customer_log )
	{

		mySqlConnection();


		setCustomer_login( customer_log );
		first_name = "";
		last_name = "";
		address = "";
		customer_password = "";
		customer_music_1 = "";
		customer_music_2 = "";
		music_1_1 = "";
		music_1_2 = "";
		music_2_1 = "";
		music_2_2 = "";
	}



	public Customer( String cus_login, String f_name, String l_name, String add, String password, String
cus_music_1, String cus_music_2 )
	{
		mySqlConnection();

		customer_login = cus_login;
```

```
        first_name = f_name;
        last_name = l_name;
        address = add;
        customer_password = password;
        customer_music_1 = cus_music_1;
        customer_music_2 = cus_music_2;
        music_1_1 = "";
        music_1_2 = "";
        music_2_1 = "";
        music_2_2 = "";
}

private void setCustomer_login( String cus_log )
{
        customer_login = cus_log;
}

private void setCustomer_id( int cus_id )
{
        customer_id = cus_id;
}

private void setFirst_name( String fis_name )
{
        first_name = fis_name;
}

private void setLast_name( String las_name )
{
        last_name = las_name;
}
```

```
private void setAddress( String add )
{
	address = add;
}

private void setPassword( String pass )
{
	customer_password = pass;
}

private void setCustomer_music_1( String cus_mus_1 )
{
	customer_music_1 = cus_mus_1;
}

private void setCustomer_music_2( String cus_mus_2 )
{
	customer_music_2 = cus_mus_2;
}

private void setMusic_1_1( String mus_1_1 )
{
	music_1_1 = mus_1_1;
}

private void setMusic_1_2( String mus_1_2 )
{
	music_1_2 = mus_1_2;
}
```

```
	private void setMusic_2_1( String mus_2_1 )
	{
		music_2_1 = mus_2_1;
	}

	private void setMusic_2_2( String mus_2_2 )
	{
		music_2_2 = mus_2_2;
	}

////////////////////////////////////////////////////////////////////////////////

	public String getCustomer_login()
	{
		return customer_login;
	}

	public int getCustomer_id()
	{
		return customer_id;
	}

	public String getFirst_name()
	{
		return first_name;
	}

	public String getLast_name()
	{
		return last_name;
	}
```

```
public String getAddress()
{
	return address;
}

public String getPassword()
{
	return customer_password;
}

public String getCustomer_music_1()
{
	return customer_music_1;
}

public String getCustomer_music_2()
{
	return customer_music_2;
}

public String getMusic_1_1()
{
	return music_1_1;
}

public String  getMusic_1_2()
{
	return music_1_2;
}
```

```
	public String getMusic_2_1()
	{
		return music_2_1;
	}

	public String getMusic_2_2()
	{
		return music_2_2;
	}

////////////////////////////////////////////////////////////////////////////////////////////////
	public boolean initCustomer()
	{
		try
		{
		// Load the JDBC driver

			//Query customer table based on customer_login variable
			String query_customer_login = "SELECT * FROM customer WHERE customer_login =
?";
			PreparedStatement pst_query_customer_login = connection.prepareStatement(
query_customer_login );
			pst_query_customer_login.setString( 1, getCustomer_login() );

			ResultSet rs_login = pst_query_customer_login.executeQuery();

			//Setup querys for customer music perfference 1 DO NOT EXECUTE
			String query_customer_music_1 = "SELECT customer_music_1, music_1, music_2
FROM customer_music INNER JOIN music_perfference ON customer_music.customer_music_1 =
music_perfference.customer_music WHERE customer_id = ?";
```

```
                PreparedStatement pst_query_customer_music_1 = connection.prepareStatement(
query_customer_music_1 );

                //Setup querys for customer music perfference 2 DO NOT EXECUTE
                String query_customer_music_2 = "SELECT customer_music_2, music_1, music_2
FROM customer_music INNER JOIN music_perfference ON customer_music.customer_music_2 =
music_perfference.customer_music WHERE customer_id = ?";
                PreparedStatement pst_query_customer_music_2 = connection.prepareStatement(
query_customer_music_2 );



                while( rs_login.next() )
                {
                        setCustomer_id( rs_login.getInt( 1 ) );
                        setCustomer_login( rs_login.getString( 2 ) );
                        setFirst_name( rs_login.getString( 3 ) );
                        setLast_name( rs_login.getString( 4 ) );
                        setAddress( rs_login.getString( 5 ) );
                        setPassword( rs_login.getString( 6 ) );

                        //setCookie_no( rs_login.getInt( 7 ) );
                        //setCookie_value( rs_login.getString( 8 ) );
                }

                if( getCustomer_id() == 0 )
                return false;

                //Execute muisc 1 and 2 querys
                pst_query_customer_music_1.setInt( 1, getCustomer_id() );
                ResultSet rs_customer_music_1 = pst_query_customer_music_1.executeQuery();
```

```
                while( rs_customer_music_1.next() )
                {
                        setCustomer_music_1( rs_customer_music_1.getString( 1 ) );
                        setMusic_1_1( rs_customer_music_1.getString( 2 ) );

                        setMusic_1_2( rs_customer_music_1.getString( 3 ) );
                }

                pst_query_customer_music_2.setInt( 1, getCustomer_id() );
                ResultSet rs_customer_music_2 = pst_query_customer_music_2.executeQuery();
                while( rs_customer_music_2.next() )
                {
                        setCustomer_music_2( rs_customer_music_2.getString( 1 ) );
                        setMusic_2_1( rs_customer_music_2.getString( 2 ) );
                        setMusic_2_2( rs_customer_music_2.getString( 3 ) );
                }



        }
        catch (SQLException e)
        {
        System.out.println( "Error 2 db" );
        }

        return true;
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    public void insertCustomer()
```

```
        {
                //insert should insert values and set all bean values, however music pefference 1_1 1_2 2_1
2_1 are not set,
                //write code to retrive these from  db and set them, write a method to do this as update will
require this too!!!!

                try
                {

                String insert_customer = "INSERT INTO customer( customer_id, customer_login,
first_name,last_name,address,password) VALUES(?,?,?,?,?,?)";
                String insert_customer_music = "INSERT INTO customer_music( customer_music_1,
customer_music_2, customer_id ) VALUES(?,?,?)";
                String get_customer_id = "SELECT customer_id FROM customer WHERE customer_login = ?";


                PreparedStatement pst_customer = connection.prepareStatement( insert_customer );
//exception
                PreparedStatement pst_customer_music = connection.prepareStatement(
insert_customer_music );
                PreparedStatement pst_get_customer_id = connection.prepareStatement( get_customer_id );


                ////////////////////////////////////////Execute customer deails inserting /////////////////

                pst_customer.setInt( 1, 0 );
                pst_customer.setString( 2, getCustomer_login() );
                pst_customer.setString(3, first_name );
                pst_customer.setString(4, last_name );
                pst_customer.setString( 5, address );
                pst_customer.setString( 6, customer_password );
```

```
            pst_customer.executeUpdate();
            pst_customer.close();
            ////////////////////////////////////////////////////////////////////////////////////////

            ////////////////////////////////GEt customer_id from db ////////////////////
            pst_get_customer_id.setString( 1, getCustomer_login() );
            ResultSet rs_customer_id = pst_get_customer_id.executeQuery();
            while( rs_customer_id.next() )
            {
                    setCustomer_id( rs_customer_id.getInt( 1 ) );
            }
            ////////////////////////////////////////////////////////////////////////////


            ////////////////////////////////////Execute music query/////////////////////////////////////

            pst_customer_music.setString( 1, customer_music_1 );
            pst_customer_music.setString( 2, customer_music_2 );
            pst_customer_music.setInt( 3, getCustomer_id() );

            pst_customer_music.executeUpdate();
            ////////////////////////////////////////////////////////////////
            }
            catch( SQLException sql )
            {
            System.out.println( sql.toString() + " In insert" );
            }
    }


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
	public void updateCustomer( String cus_login, String f_name, String l_name, String add, String
password, String cus_music_1, String cus_music_2 )
	{

		setCustomer_login( cus_login );
		setFirst_name( f_name );
		setLast_name( l_name );
		setAddress( add );
		setPassword( password );
		setCustomer_music_1( cus_music_1 );
		setCustomer_music_2( cus_music_2 );

		try
		{
		String update_customer = "UPDATE customer SET customer_login = ?, first_name = ?,
last_name = ?, address = ?, password = ? WHERE customer_id = ?";
		PreparedStatement pst_upadte_customer = connection.prepareStatement( update_customer );

		pst_upadte_customer.setString( 1, getCustomer_login() );
		pst_upadte_customer.setString( 2, getFirst_name() );
		pst_upadte_customer.setString( 3, getLast_name() );
		pst_upadte_customer.setString( 4, getAddress() );
		pst_upadte_customer.setString( 5, getPassword() );
		pst_upadte_customer.setInt( 6, getCustomer_id() );

		pst_upadte_customer.executeUpdate();
		}
		catch( SQLException sql )
		{
			System.out.println( sql.toString() + " In update seg 1" );
```

```
		}


		try
		{
		String update_customer_music = "UPDATE customer_music SET customer_music_1 = ?,
customer_music_2 = ? WHERE customer_id = ?";
		PreparedStatement pst_upadte_customer_music = connection.prepareStatement(
update_customer_music );

		pst_upadte_customer_music.setString( 1, getCustomer_music_1() );
		pst_upadte_customer_music.setString( 2, getCustomer_music_2() );
		pst_upadte_customer_music.setInt( 3, getCustomer_id() );

		pst_upadte_customer_music.executeUpdate();
		}
		catch( SQLException sql )
		{
			System.out.println( sql.toString() + " In update seg 2" );
		}
	}

	/*public String getLogin()
	{
		try
		{
			String query_customer_login = "SELECT password FROM customer WHERE
customer_login = ?";
			PreparedStatement pst_query_customer_login;

			pst_query_customer_login = connection.prepareStatement( query_customer_login );
```

```
                String login_id = "kumar_nirmal@hotmail.net";
                pst_query_customer_login.setString( 1, login_id );
                ResultSet rs = pst_query_customer_login.executeQuery();

                //String db_password = "empty";
                //String db2_password = "empty2";

                while( rs.next() )
                {
                        db_password = rs.getString( 1 );
                }

                return db_password;
        }
        catch( SQLException sql )
        {
                System.out.println( sql.toString() + " In update seg 2" );
        }
        return null;

} */



private void mySqlConnection()
{
        try
        {
        Class.forName( driverName );
```

```
                // Create a connection to the database
                connection = DriverManager.getConnection(url, username, password);

                System.out.println( "conncetion made... " );

                }
                catch (ClassNotFoundException e)
                {
                System.out.println( "Error 1 driver " + e.toString() );
                }
                catch (SQLException e)
                {
                System.out.println( "Error 2 db" );
                }

        }


        public static void main( String args[] )
        {

                /*Customer cus = new Customer( "kumar@hotmail.com", "Nirmal", "Kumar", "Wembley", "java",
"house", "techno" );

                cus.insertCustomer();

                if( cus.initCustomer() )
                System.out.println( "NAme: " + cus.getFirst_name() + " Second: " + cus.getLast_name() + " Cus
id " + cus.getCustomer_id() + " add " + cus.getAddress() + " mus1 " + cus.getCustomer_music_1() + " mus11 "
```

```
+ cus.getMusic_1_1() + " mus12 " + cus.getMusic_1_2() + " mus2 " + cus.getCustomer_music_2() + " mus21 "
+ cus.getMusic_2_1() + " mus22 " + cus.getMusic_2_2());
                    else
                    System.out.println( "Customer not found" );



                    Customer c = new Customer( "kumar@hotmail.com" );

                    if( c.initCustomer() )
                    System.out.println( "NAme: " + c.getFirst_name() + " Second: " + c.getLast_name() + " Cus id "
+ c.getCustomer_id() + " add " + c.getAddress() + " mus1 " + c.getCustomer_music_1() + " mus11 " +
c.getMusic_1_1() + " mus12 " + c.getMusic_1_2() + " mus2 " + c.getCustomer_music_2() + " mus21 " +
c.getMusic_2_1() + " mus22 " + c.getMusic_2_2());
                    else
                    System.out.println( "Customer not found" );



                    c.updateCustomer( "kumar_nirmal@hotmail.net", "Nirmal", "Raj", "Farridon", "xml", "latin", "soul"
);
                    System.out.println( "NAme: " + c.getFirst_name() + " Second: " + c.getLast_name() + " Cus id "
+ c.getCustomer_id() + " add " + c.getAddress() + " mus1 " + c.getCustomer_music_1() + " mus11 " +
c.getMusic_1_1() + " mus12 " + c.getMusic_1_2() + " mus2 " + c.getCustomer_music_2() + " mus21 " +
c.getMusic_2_1() + " mus22 " + c.getMusic_2_2());
                    */


                    //Customer c = new Customer( "kumar_nirmal@hotmail.net" );
                    //System.out.println( "login pass " + c.getLogin() );
```

```
        }
    }
```

Registration.java

```
package handler;

import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import controller.*;


public class Registration extends HttpServlet
{
	Customer c;
	HttpSession session;
	RequestDispatcher dispatch, noSession;

	String first_name = "";
	String last_name = "";
	String address = "";
	String customer_login = "";
	String password = "";
	String customer_music_1 = "";
	String customer_music_2 = "";

	public void doGet( HttpServletRequest request, HttpServletResponse response ) throws
ServletException, IOException
	{

////////////////////////////////////Variables////////////////////////////////////////////////

	noSession = getServletContext().getRequestDispatcher("/checker");
	dispatch = getServletContext().getRequestDispatcher("/locate");
```

```
//////////////////////////////////////Check for session/////////////////////////////////////////////////////

        session = request.getSession( false );

        //Test whether user has enter page without going via authentictor if so redirest to authentictor
        if( session == null )
        {
                //request.setAttribute( "target", Integer.toString( Page.INDEX ) )
                noSession.forward( request, response );
        }


        String browserSessionString = (String) session.getAttribute( "browser" );
        int browserSession = Integer.parseInt( browserSessionString );
//////////////////////////////////////Begin assign form parameter to variables    ////////////////////////////


        customer_login = request.getParameter( "input_user_id" );

        first_name = request.getParameter( "input_first_name" );
        last_name = request.getParameter( "input_last_name" );
        address = request.getParameter( "input_address" );
        password = request.getParameter( "input_password_id" );


        if( browserSession == Device.PC )
        {
                customer_music_1 = request.getParameter( "music_1" );
                customer_music_2 = request.getParameter( "music_2" );
        }
```

```
        else if( browserSession == Device.WAP )
        {
                String wml_music = request.getParameter( "music_type" );
                //More coding required to separte the String
        }

//////////////////////////////////////////Create bean add to session  ////////////////////////////////////////////////////////////////



                /*response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                String docType =
                 "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
                 "Transitional//EN\">\n";
                out.println(docType +
            "<HTML>\n" +
            "<HEAD><TITLE>" + customer_login + " " + first_name + " </TITLE></HEAD>\n" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1> db password is " + customer_login + " " + first_name + last_name + address +
password +  "</H1>\n" +
            "</BODY></HTML>");
                 */



        c = new Customer( customer_login, first_name, last_name, address, password,
customer_music_1, customer_music_2 );
        c.insertCustomer(); // exception
```

```
        session.setAttribute( "bean", c );

////////////////////////////////////////Deploy cookie to client ////////////////////////////////////////////////

        Cookie mmuser = new Cookie("mmuser", c.getCustomer_login() );
        Cookie mmpwd = new Cookie("mmpwd",c.getPassword() );

        response.addCookie(mmuser);
        response.addCookie(mmpwd);

        request.setAttribute( "target", Integer.toString( Page.MUSIC ) );
        dispatch.forward( request, response );
////////////////////////////////////////////////////////////////////

        }
}
```

Login.java

```
package handler;

import java.sql.*;
import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import controller.Page;
//import handler.Customer;


public class Login extends HttpServlet
{
	Connection connection;
	PreparedStatement pst_query_customer_login;
	RequestDispatcher dispatch, noSession;

	String db_password = "";

	String serverName = "localhost";
      String mydatabase = "music";
      String url = "jdbc:mysql://" + serverName +  "/" + mydatabase; // a JDBC url
      String username = "admin";
      String password = "jdbc";
	String driverName = "org.gjt.mm.mysql.Driver"; // MySQL MM JDBC driver

	//dispatch = getServletContext().getRequestDispatcher("/locate");


	boolean validLogin = false;
	HttpSession session;
```

```
	public void init()
	{
		try
		{

		mySqlConnection();

		String query_customer_login = "SELECT password FROM customer WHERE customer_login =
?";

		pst_query_customer_login = connection.prepareStatement( query_customer_login );

		dispatch = getServletContext().getRequestDispatcher("/locate");

		}
		catch( SQLException se)
		{
			System.out.println("Error creating PreparedStatement : "+ se );
		}
		catch( NullPointerException npe )
		{
			System.out.println("Error creating PreparedStatement npe: "+ npe );
		}
	}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
	private void mySqlConnection()
	{
		try
		{
```

```
                Class.forName( driverName );

                // Create a connection to the database
                connection = DriverManager.getConnection(url, username, password);
                }
                catch (ClassNotFoundException e)
                {
                System.out.println( "Error 1 driver" );
                }
                        catch (SQLException e)
                {
                System.out.println( "Error 2 db" );
                }

        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        public void doGet( HttpServletRequest request, HttpServletResponse response ) throws
ServletException, IOException
        {
                //dispatch = getServletContext().getRequestDispatcher("/locate");
                noSession = getServletContext().getRequestDispatcher("/checker");

                session = request.getSession( false );

                //Test whether user has enter page without going via authentictor if so redirest to authentictor
                if( session == null )
                {
                        //request.setAttribute( "target", Integer.toString( Page.LOGIN ) )
                        noSession.forward( request, response );
```

```
                }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                String cookStatus = ( String ) session.getAttribute( "user_id" );

                //Test whether cookie values are avaliable if so prompt automatic login

                //NO cookie data aviable therefore, get form parameter values
                if( cookStatus == null )
                {
                        String login = request.getParameter( "input_user_id" );

                        String no_cook_password = request.getParameter( "input_password_id" );
                        //System.out.println( login + " " + password );
                        validLogin = validateLogin( login, no_cook_password,  request, response  );
                }
                //Cookie data avilable prompt auto login. cookStatus is the user_id
                else if( cookStatus != null )
                {
                        validLogin = validateLogin( cookStatus, ( String ) session.getAttribute( "password_id" ),
request, response  );
                }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                //Test whether login is valid
                if( validLogin )
                {
                        request.setAttribute( "target", Integer.toString( Page.MUSIC ) );
                }
```

```
                else
                {
                        request.setAttribute( "target", Integer.toString( Page.INDEX ) );
                }

                dispatch.forward( request, response );
        }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

        private boolean validateLogin( String login_id, String password_id, HttpServletRequest request,
HttpServletResponse response  ) throws ServletException, IOException
        {
                try
                {
                pst_query_customer_login.setString( 1, login_id );
                ResultSet rs = pst_query_customer_login.executeQuery();

                //String db_password;

                //String db2_password = "empty2";

                while( rs.next() )
                {
                        db_password = rs.getString( 1 );
                }

                //String temp = getLogin();

                /*
                response.setContentType("text/html");
```

```
            PrintWriter out = response.getWriter();
            String docType =
             "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
             "Transitional//EN\">\n";
            out.println(docType +
          "<HTML>\n" +
          "<HEAD><TITLE>" + db_password + " " + login_id + "</TITLE></HEAD>\n" +
          "<BODY BGCOLOR=\"#FDF5E6\">\n" +
          "<H1> db password is " + db_password + " " + login_id + " " + password_id + " " +"</H1>\n" +
          "</BODY></HTML>");
            */

            if( db_password.equalsIgnoreCase( password_id ) )//Documentation ignore case dont trouble
WAP user
            {

                  //Create Java bean, customer bean

                  Customer c = new Customer( login_id );

                  c.initCustomer();

                  session.setAttribute( "bean", c );

                  //MusicBean mb = new MusicBean();
                  //mb.initialize(login_id);
                  //session.setAttribute( mb );
                  return true;
            }
            }
            catch( SQLException sql )
```

```
		{
			System.out.println( "SQL error" + sql );
		}
		return false;
	}


	public String getLogin()
	{
		try
		{

			String query_customer_login = "SELECT password FROM customer WHERE
customer_login = ?";
			PreparedStatement pst_query_customer_login;

			pst_query_customer_login = connection.prepareStatement( query_customer_login );

			String login_id = "kumar_nirmal@hotmail.net";
			pst_query_customer_login.setString( 1, login_id );
			ResultSet rs = pst_query_customer_login.executeQuery();

			//String db_password = "empty";
			//String db2_password = "empty2";

			while( rs.next() )
			{
				db_password = rs.getString( 1 );
			}

			return db_password;
```

```
                }
                catch( SQLException sql )
                {
                        System.out.println( sql.toString() + " In update seg 2" );
                }
                return null;

        }



}


        /*
                        Try retrieving the predefined cookie variables (eg. mmuser,mmpwd)

                        check if it is null
                        if null, redirect the user to login
                        else, redirect the user to music page

                        -----> Storing a cookie in the client's machine after successful login

                        Cookie mmuser = new Cookie("mmuser",login);
                        Cookie mmpwd = new Cookie("mmpwd",password);

                        response.addCookie(mmuser);
                        response.addCookie(mmpwd);

                        ----> retrieval
```

```
Cookies mmusiccookies[] = request.getCookies();
String cook_login = "";
String cook_pwd = "";
boolean cook_login_ok = false;
boolean cook_pwd_ok = false;

for (int i=0;i<mmusiccookies.length;i++)
{
	String name = mmusiccookies[i].getName();

	if(name.equalsIgnoreCase("mmuser"))
	{
		cook_login = mmusiccookies[i].getValue();
		cook_login_ok = true;

	}
	if(name.equalsIgnoreCase("mmpwd"))
	{
		cook_pwd = mmusiccookies[i].getValue();
		cook_pwd_ok = true;
	}

	if(cook_login_ok && cook_pwd_ok)
		break;
}
```

Profile.java Incomplete

```
package handler;

import java.text.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import controller.Device;


public class Profile extends HttpServlet
{
	Customer c;

	HttpSession session;
	RequestDispatcher dispatch, noSession;

	noSession = getServletContext().getRequestDispatcher("/checker");
	dispatch = getServletContext().getRequestDispatcher("/xslt");

	String customer_music_1 = "";
	String customer_music_2 = "";
	String music_1_1 = "";
	String music_1_2 = "";
	String music_2_1 = "";
	String music_2_2 = "";

	public void doGet( HttpServletRequest request, HttpServletResponse response ) throws
ServletException, IOException
	{

//////////////////////////////////////Check for session/////////////////////////////////////////////////
```

```
        session = request.getSession( false );

        //Test whether user has enter page without going via authentictor if so redirest to authentictor
        if( session == null )
        {
                //request.setAttribute( "target", Integer.toString( Page.LOGIN ) )
                noSession.forward( request, response );
        }

////////////////////////////////////Get customer bean/////////////////////////////////////////////////////////

        c = ( Customer ) session.geAttribute( "bean" );

        customer_music_1 = c.getCustomer_music_1();
        customer_music_2 = c.getCustomer_music_2();
        music_1_1 = c.getMusic_1_1();
        music_1_2 = c.getMusic_1_2();
        music_2_1 = c.getMusic_2_1();
        music_2_2 = c.getMusic_2_2();

////////////////////////////////////Check request and redirect//////////////////////////

        int request = request.getParameter( "requesttype" );
        String music_request = "";

        if( request == 1 )
        {
                music_request = customer_music_1;
        }
        else if( request == 2 )
```

```
{
	music_request = customer_music_2;
}
else if( request == 3 )
{
	music_request = music_1_1;
}
else if( request == 4 )
{
	music_request = music_1_2;
}
else if( request == 5 )
{
	music_request = music_2_1;
}
else if( request == 6 )
{
	music_request = music_2_2;
}


if( music_request
```

JSPs

profile.jsp

```
<%@ page language="java" import="controller.*,handler.*" %>

<%
	String first_name = "";
	String music_1 = "";
	String music_2 = "";
	String music_1_1 = "";
	String music_1_2 = "";
	String music_2_1 = "";
	String music_2_2 = "";



	RequestDispatcher dispatch = getServletContext().getRequestDispatcher("/checker");

	session = request.getSession( false );

	//Test whether user has enter page without going via authentictor if so redirest to authentictor
	if( session == null )
	{
		//request.setAttribute( "target", Integer.toString( Page.LOGIN ) )
		dispatch.forward( request, response );
	}


	//Customer c = new Customer( "vsakaria@hotmail.com", "vishal", "sakaria", "6 alan", "test", "techno",
"house" );

	Customer c;
	c = (Customer) session.getAttribute( "bean" );
```

```
	first_name = c.getFirst_name();
	music_1 = c.getCustomer_music_1();
	music_2 = c.getCustomer_music_2();
	music_1_1 = c.getMusic_1_1();
	music_1_2 = c.getMusic_1_2();
	music_2_1 = c.getMusic_2_1();
	music_2_2 = c.getMusic_2_2();



	//Get browser type from session convert to int
	String browserSessionString = (String) session.getAttribute( "browser" );
	int browserSession = Integer.parseInt( browserSessionString );

	//int browserSession = Device.PC;


	if( browserSession == Device.PC )
	{
		response.setContentType( "text/html" );



%>
<html>
<head>
	<title>M-Muisc personalised page for <%= first_name %> </title>

	<SCRIPT LANGUAGE="javascript">
```

```
	function check( type )
	{

		var request =  type;
		alert( "type is " +type );
		alert( "request is set to " + request )

		document.music.requesttype.value = request;

		document.music.submit()


		var request2 = document.music.requesttype.value;

		alert( "request type is set to " + request2 )
	}

	</script>




</head>

<body>
	<H1> Welcome <%= first_name %> to your personalised music page! </H1>
	<br/>

	<p>
	<h2>
	Below is a set of links personalised for you, The first two music genre are choosen by you
```

```
        the remaining genres are set by our database. Enjoy!
        </h2>
        </p>

        <FORM name ="music" METHOD="GET" ACTION="">

        <input type="hidden" name="requesttype" />

        <table>
        <tr>

                <td><INPUT TYPE="submit" VALUE="<%= music_1%>" Name="<%= music_1%>"
onClick="check( 1 )"></td>
                <td><INPUT TYPE="submit" VALUE="<%= music_2%>" Name="<%= music_2%>"
onClick="check( 2 )"></td>
                <td><INPUT TYPE="submit" VALUE="<%= music_1_1%>" Name="<%= music_1_1%>"
onClick="check( 3 )"></td>
                <td><INPUT TYPE="submit" VALUE="<%= music_1_2%>" Name="<%= music_1_2%>"
onClick="check( 4 )"></td>
                <td><INPUT TYPE="submit" VALUE="<%= music_2_1%>" Name="<%= music_2_1%>"
onClick="check( 5 )"></td>
                <td><INPUT TYPE="submit" VALUE="<%= music_2_2%>" Name="<%= music_2_2%>"
onClick="check( 6 )"></td>
        </tr>
        </table>

        </form>
</body>
</html>
```

```
<%

        }
        else if( browserSession == Device.WAP )
        {
                response.setContentType("text/vnd.wap.wml");
%>

        <?xml version="1.0"?>
        <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="music_profile" title="Personalised music">

        <!--<do type="options" label="<%= music_1 %>">
                        <go href = "#login"/>
        </do>

        <do type="options" label="<%= music_2 %>">
                <go href = "#registration"/>
        </do>-->



   <p>
        Welcome <%= first_name %>
        Click to display your personalised music options
   </p>

        </card>
```

```
</wml>
<%
        }

%>
```

Index.jsp

```
<%@ page language="java" import="controller.*" %>

<%
	response.setContentType("text/vnd.wap.wml");
	RequestDispatcher dispatch = getServletContext().getRequestDispatcher("/checker");
	dispatch.forward(request,response);


%>
```

XML and XSL Files

Index.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<welcome>
	<image>m-muisc.gif</image>
	<!-- What do i do about images -->
	<title_bar>Welcome to M-Music</title_bar>
	<title>Welcome to M-music</title>
	<!-- I want include wired or wireless site -->
	<info_1>
	M-Muisc is a web site dedicated to muisc. The difference is you can access inforamtion on the move.
	</info_1>
	<info_2>
	Setup by Vishal Sakaria for his 3rd year project at the London GuildHall University, this web site is written using
Java and XML  Technology.  The Java side inculdes Java Server Pages, Java Servlets and Java Beans, using Apache
Tomat version 4.1.18 the  website consists of all  standards supported by Sun Microsystems and the Jakarta
Project. XML is used to create the base content  files, such as this content,  which are platform, operator system and
device indepentent, this allows us to send cotent to a wide  range of devices such as a WAP  enabled phone.
XSL and XSLT are used to transform the XML files into the correct format, such  as WML for a  WAP device.
	</info_2>
	<info_3>
	The web site is wholey dedicded to DANCE muisc it includes House,Techno, Trance, Funky
	Soulful, Latin, Big Beat and Hard House.
	</info_3>
	<wap_info>
	Use the "option" button to navigate throught the site. Enjoy!
	</wap_info>
	<wap_info_2>
	Created by Vishal Sakaria using Java and XML
	</wap_info_2>
	<status>This area Describes the status of the devices cookie settings</status>
```

```
        <link_1 name="M-Login"/>
        <!--create link vlaues-->
        <link_2 name="M-Registration"/>
</welcome>
```

Registration.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<registration>
	<image>Some image</image>
	<title>M-Registration for New User</title>
	<title_bar>M-Registration</title_bar>
	<info_1>
	This is a personalised registration form, choose your muisc preference and m-muisc will display your
personalised page including your
	preferences plus other similar music genres!
	</info_1>
	<input_user>
		<input_first_name type="text">First Name</input_first_name>
		<input_last_name type="text">Last Name</input_last_name>
		<input_address type="text">Address</input_address>
		<input_user_id type="text">E_mail Address (user id)</input_user_id>
		<input_password_id type="password" size="12" maxlength="10">Password (10
Charactors)</input_password_id>
		<input_retype_id type="password" size="12" maxlength="10">Re-type
password</input_retype_id>
	</input_user>

	<input_music>
		<preference_1 name="house">House</preference_1>
		<preference_2 name="techno">Techno</preference_2>
		<preference_3 name="trance">Trance</preference_3>
		<preference_4 name="big beat">Big Beat</preference_4>
		<preference_5 name="soul">Soulful</preference_5>
		<preference_6 name="hard house">Hard House</preference_6>
		<preference_7 name="funk">Funk</preference_7>
```

```
                <preference_8 name="latin">Latin</preference_8>
        </input_music>
        <!--i think the muisc types should be embedded in these tags-->
        <link_1>M-Submit</link_1>
        <wap_link>M-Login</wap_link>
</registration>
```

Login.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<login>
	<image>some image</image>
	<title_bar>M-Login</title_bar>
	<title>M-Login for Existing User</title>
	<info_1>
		Login to m-music to view your personlised page, if you have forgotten your password click on
the forgotten button,
		otherwise click Login.
	</info_1>
	<wap_info>Login to m-music to view your personlised page</wap_info>
	<input_user_id>Input your User ID</input_user_id>
	<input_password_id>Input your password</input_password_id>


	<link_1>Forgotten your password</link_1>
	<link_2>Login</link_2>
</login>
```

house.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<music_profile type="House" wap_type="house">
	<image>Some image</image>
	<title>M-Muisc House profile</title>
	<defenition>
		<muisc_defenition>
		House typically runs around 120 beats per minute (BPM) and uses a 4/4 time sequence and an eight bar repeating cycle.
		</muisc_defenition>
		<music_defenition_extra>
		The beat is maintained by a heavy kick drum that alternates in a 1-3 pattern with a high hat accent. Like its disco roots, house often features vocal choruses, real instrumentation, and a more traditional song structure.
	</music_defenition_extra>
	</defenition>

	<timeline>
		<line year="1985">
		Like it or not, house was first and foremost a direct descendant of disco. Disco had already been going for ten years when the first electronic drum tracks began to appear out of Chicago


		</line>
		<line_extra>
		and in that time it had already suffered the slings and arrows of merciless commercial exploitation, dilution and racial and sexual prejudice which culminated in the 'disco sucks' campaign. In one bizarrely extreme incident, people attending a baseball game in Chicago's Komishi Park were invited to bring all their unwanted disco records and after the game they were tossed onto a massive bonfire. Disco eventually collapsed under a heaving weight of crass disco versions of pop records and
```

an ever- increasing volume of records that were simply no good. But the underground scene had already stepped off and was beginning to develop a new style that was deeper, rawer and more designed to make people dance. Disco had already produced the first records to be aimed specifically at DJs with extended 12" versions that included long percussion breaks for mixing purposes and the early eighties proved a vital turning point. Sinnamon's 'Thanks To You', D-Train's 'You're The One For Me' and The Peech Boys' 'Don't Make Me Wait', a record that's been continually sampled over the last decade, took things in a different direction with their sparse, synthesized sounds that introduced dub effects and drop-outs that had never been heard before.

</line_extra>

<line year="1986">

While Frankie Knuckles had laid the groundwork for house at the Warehouse, it was to be another DJ from the gay scene that was really to create the environment for the house explosion - Ron Hardy. Where Knuckles' sound was still very much based in disco, Hardy was the DJ that went for the rawest, wildest rhythm tracks he could find and he made

</line>

<line_extra>

The Music Box the inspirational temple for pretty much every DJ and producer that was to come out of the Chicago scene. He was also the DJ to whom the producers took their very latest tracks so they could test the reaction on the dance floor. Larry Heard was one of those people.

"People would bring their tracks on tape and the DJ would play spin them in. It was part of the ritual, you'd take the tape and see the crowd reaction. I never got the chance to take my own stuff because Robert (Owens) would always get there first."

"The Music Box was underground " remembers Adonis. "You could go there in the middle of the winter and it'd be as hot as hell, people would be walking around with their shirts off. Ron Hardy had so much power people would be praising his name while he was playing, and I've got the tapes to prove it!

"The difference between Frankie and Ronnie was that people weren't making records when Frankie was playing, though all the guys who would become the next DJs were there checking him out. It was The Music Box that really inspired people. I went there one night and the next day I was in the studio making 'No Way Back' " In 1985 the records were few and far between. By 1986 the trickle had turned to a flood and it seemed like everybody in Chicago was making house music. The early players were joined by a rush of new talent which included the first real vocal talents of house - Liz Torres, Keith Nunally who worked with Steve Hurley, and Robert Owens who joined up with Larry Heard to form Fingers Inc, though the duo had already worked with Harri Dennis on The It's 'Donnie' -and key producers like Adonis, Mr Lee, K Alexi and a guy who was developing a deep, melodic sound that relied on big strings and pounding piano - Marshall Jefferson.

Marshall worked with a number of people like Harri Dennis and Vince Lawrence for projects like Jungle Wonz and Virgo, who made the stunning 'RU Hot Enough'. But it was 'Move Your Body' that became THE house record of 1986, so big that both Trax and DJ International found a way to release it, and it was no idle boast when the track was subtitled 'The House Music Anthem', because that's exactly what it was. Jefferson was to become the undisputed king of house, going on to make a string of brilliant records with Hercules and On The House and developing the quintessential deep house sound first with vocalist Curtis McClean and then with Ce Ce Rogers and Ten City. "I can remember clearing a floor with that record" laughs J azzy M. "Though they'd started playing it in Manchester, most of London was still caught up in that rare groove and hip hop thing. A lot of people were saying to me 'why are you playing this hi- NRG' and it was hard work but people were starting to get into it." 'Move Your Body' was undoubtedly the record that really kicked off house in the UK, first played repeatedly by the established pirate radio stations in London, which at the time played right across the Black music spectrum, and then by club DJs like Mike Pickering, Colin Faver, Eddie Richards, Mark Moore and Noel and Maurice Watson, the latter two playing at the first club in London to really support house - Delirium.

</line_extra>

<line year="1987">

While Chicago stole the thunder in 1986, other cities not only in the United States but across the world had either been absorbing house or working on their own thing, biding their time.

</line>
<line_extra>
One record from New York served a warning shot that the city was gearing up for some serious action - 'Do It Properly' by 2 Puerto Ricans, A Blackman and A Dominican. 'Do It Properly' was essentially a bootleg of Adonis' 'No Way Back' with loads of samples and a great electronic keyboard riff squeezed in to it and the first in a long, long line of New York sample house tracks. Its producers were one Robert Clivilles and David Cole, helped by another guy called David Morales. After that some kid in Brooklyn called Todd Terry made a couple of sample tracks with a freestyle groove for Fourth Floor Records by an act he called Masters At Work.

But the sound that was really taking shape in New York and New Jersey was a deep style of club music based on a heritage that had its roots firmly in r'n'b. Though there were some superb deep, emotive instrumentats like Jump St. Man's 'B-Cause', the emphasis was on songs, which came with Arnold Jarvis' 'Take Some Time', Touch's 'Without You', Exit's 'Let's Work It Out' and a record on Movln, a new label run from a record store in New Jersey's East Orange - Park Ave's 'Don't Turn Your Love'. Ironically, as the first garage hits began to appear, The Paradise Garage - Larry Levan had already left - closed, but the vibe carried on with Blaze, who recorded 'If You Should Need A Friend' and Jomanda, both of whom teamed up with new New York label Quark.

Echoing the need for vocals in house music, deep house began to take hold in Chicago. Following Marshall Jefferson's lush productions, the record that defined deep house was the Nightwriters' 'Let The Music Use You', mixed by Frankie Knuckles and sung by Ricky Dillard, a record that a year later was to become one of the anthems of the UK's Summer Of Love. And it didn't end there. Kym Mazelle launched her career with 'Taste My Love' and 'I'm A Lover', while Ralphie Rosario unleashed the monstrous 'You Used To Hold Me' featuring the wailing tonsils of Xavier Gold. Then there was Ragtyme's 'I Can't Stay Away', sung by a guy who sounded a a little like a new Smokey Robinson - Byron Stingily. Soon after, Ragtyme, who also made an extremely silly innuendo track called 'Mr Fixit Man', mutated into Ten Clty. But Chicago's excursion into songs wasn't only characterised by uplifting wailers. There was another side, led by the weird, melanchoty songs of Fingers Inc and beginning to show itself in other minimalist productions like MK II's 'Don't Stop The Muslc' and 2 House People's 'Move My Body'. By 1987, though house was no longer a

tale of two cities. The virus was taklng hold elsewhere as clubbers, DJs and producers worldwide became exited by the new music.

</line_extra>

<line year="1988">

In truth, acid house had already started long before 1988. Amongst the scores of Chicagoans who were buying equipment and trying to learn how to make tracks was one DJ Pierre, who'd started out playing Italian imports at roller discos in the Chicago suburbs, and who had joined Lil Louis for his notorious parties.

</line>

<line_extra>

"Phuture was me and two other guys, Spanky and Herbert J." remembers Pierre. "We had this Roland 303, which was a bassline machine, and we were trying to figure out how to use it. When we switched it on, that acid sound was already in it and we liked the sound of it so we decided to add some drums and make a track with it. We gave it to Ron Hardy who started playing it straight away. In fact, the first time he played it, he played it four times in one night! The first time people were like, 'what the fuck is this?' but by the the fourth they loved it. Then I started to hear that Ron was playing some new thing they were calling 'Ron Hardy's Acid Trax', and everybody thought it was something he'd made himself. Eventually we found out that it was our track so we called it 'Acid Trax'. I think we may have made it as early as 1985, but Ron was playing it for a long time before it came out."

Explanations for the name of 'acid' have been long and varied, but the most popular, and the one endorsed by a number of people who were there at the time was that they used to put acid in the water at the Music Box. Pierre though, stresses that Phuture was always anti- drugs, and cites a track about a cocaine nightmare, 'Your Only Friend' that was on the same EP as 'Acid Trax'. 'Acid Trax' came out in 1986 but made little impact outside Chicago, as was the case with another acid track, Sleazy D's 'I've Lost Control', which slapped a deranged laugh and some geezer repeating the title over the 303 squelching. 'I've Lost Control' was made by Adonis and Marshall Jefferson and was certainly the first acid track to make it to vinyl, though which was created first will possibly never be known for sure. It wasn't until well into 1987 that the acid sound began to infiltrate Britain, fuelled by

another track that was getting a lot club play, and which fitted into the sound Bam Bam's 'Give It To Me', and a diversion of the regular acid track which put vocals into the equation, developed by Pierre's Phantasy Club with 'Fantasy Girl'. The house scene in Britain had faltered following the commercialisation of the poppier end of the spectrum, but towards the end of 1987 the underground was taking off with new LP compilation series like 'Jack Trax' and the opening in London of seminal clubs like Shoom and Spectrum and the move of Delirium to Heaven where the main dancefloor became exclusively house. Delirium's Deep House Convention atLeicester Square's Empire in February 1988 which featured a number of seminal Chicago artists like Kym Mazelle, Fingers Inc, Xavier Gold. Marshall Jefferson and Frankie Knuckles was a depressing event because of the poor turnout. But the people who did go were to be become the prime movers of London's house explosion. The next week a warehouse party called Hedonism was rammed and the soundtrack was acid. Acid house UK style had begun.

</line_extra>

<line year="1989">

By now the UK and its trend-hungry music press had become the local point of the dance music world.

</line>

<line_extra>

By now the UK and its trend-hungry music press had become the local point of the dance music world. After acid had slumped into fatuousness with the adopted logo of acid, the smiley, appearing on t- shirts racked up in every high street and the mainstream press (including the 'qualities') scuttling after every whiff of a half-arsed drug story, they discovered new beat from Belgium. The trouble was that save for one or two genuinely good records like A Split Second's 'Flesh', nearly everyone outside Belgium hated new beat, a sort of sluggish cross between acid, techno and heavy industrial Euro music and the media hype dissolved into a number of red faces.

</line_extra>

</timeline>

<general>

<general_information>

It's been ten years since the first identifiably house tracks were put on to vinyl, ten years which have changed the technology behind the electronic music revolution beyond recognition but left the basic structure of house intact.

</general_information>
<general_information_extra>
It's seven years since it was being said house couldn't last, that it was just hi-NRG, a fast blast that would wither as quickly as it had started. But then the music reinvented itself, and then again and again until it gradually dawned on people that house wasn't just another phase of club culture, it was club culture, the continuing future of dance music. The reason? It's simple. People like to dance to house.
</general_information_extra>
</general>
</music_profile>

techno.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<music_profile type="Techno" wap_type="techno">
	<image>Some image</image>
	<title>M-Muisc Techno profile</title>
	<defenition>
		<muisc_defenition>
	Techno Music: An electronic evolution of house developed in Detroit.
	</muisc_defenition>
		<music_defenition_extra>
	Techno uses the 4/4 beat structure and eight bar repeating structure of house to build
predominantly percussive tracks that use purely electronic sounds. Originally influenced by the
	European electro sounds of Kraftwerk and electronic tracks from the Italo label in Italy, techno is
using "Technology" to make muisc.
	</music_defenition_extra>
	</defenition>
	<timeline>
		<line year="1867" >
		Hipps invents the Electromechanical Piano in Neuchatel (Switzerland).
		</line>
		<line year="1876" >
		Elisha Gray invents the Electroharmonic (an instrument that transmitted musical tones over
wires). She was also the inventer of the telephone along with A. Graham Bell.
		</line>
		<line year="1877" >
		Thomas Edison invents the Phonograph.
		</line>
		<line year="1880" >
		Alexander Graham Bell invenstigates and patent several ways for transmitting and recording
sound in laboratory in Washington, D.C. along with Charles S. Tainter.
```

</line>
<line year="1893" >
Eric Satie composes "Vexations" the first loop compositions for piano
</line>
<line year="1895" >
Julian Carillo starts the construction of a serie of instruments to reproduce divisions as small as a sixteenth tone, including the Octavina for eighth tones and an Arpa Citera for sixteenth tones based in his theories of microtones, 96 tone scale.
</line>
<line year="1897" >
E.S. Votey invents the Pianola, an instrument that used a pre-punched, perforated paper roll moved over a capillary bridge.
</line>
<line year="1898" >
Valdemar Poulson patents the Telegraphone -the first magnetic recording machine.
</line>
</timeline>
<general_information>
Techno had originated in Detroit in the mid-'80s. At that time techno was strictly electronic music while house still had tight connection to disco.
<general_information_extra>
DJs Kevin Saunderson, Juan Atkins, and Derrick May pioneered techno music emphasizing the electronic, synthesized beats of electro-funk artists like Afrika Bambaataa and Kraftwerk. In the US, techno was strictly an underground phenomenon, but in UK, it broke into the mainstream in the late '80s. In the early '90s, techno began to fragment into a number of subgenres, including hardcore, ambient, and jungle.
</general_information_extra>
</general_information>
</music_profile>

Index_HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>


<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<!--Welcome StyleSheet HTML page - index.xml -->
<!--Templete caller -->
<xsl:template match="/">

<html>

<head>
<title><xsl:call-template name="title_bar"/></title>

	<script language="javascript">

	function check( type )
	{
		var request =  type;

		<!--
		alert( "type is " + type );
		alert( "request is set to " + request )
		-->

		document.request.requesttype.value = request;

		document.request.submit()
```

```
                <!--
                var request2 = document.request.requesttype.value;

                alert( "request type is set to " + request2 )
                -->
        }

        </script>
</head>

<body>

<center>

<h1><b><xsl:call-template name="title"/></b></h1>

<br/>
<br/>

<img src="C:\apache\Tomcat 4.1\webapps\mmusic\images\m-music.gif" align="middle"/>

<br/>
<br/>
<br/>
<br/>

<h2><xsl:call-template name="information1"/></h2>

<p><h4><xsl:call-template name="information2"/></h4></p>

<h3><xsl:call-template name="information3"/></h3>
```

```
<br/>
<br/>
<br/>


<table>
	<tbody>
		<tr>
			<form name="request" method="get" action="/mmusic/checker">

			<input type="hidden" name="requesttype" />

			<td><xsl:call-template name="link1"/></td>
			<td><xsl:call-template name="link2"/></td>

			</form>
		</tr>
	</tbody>
</table>

</center>

</body>
</html>
<!--End templete call segment-->




</xsl:template>
```

```
<!--Begin templete definition -->

<!-- Display Title bar-->
<xsl:template  name="title_bar" >

        <xsl:for-each select="/welcome/title_bar" >

            <xsl:value-of select="."/>

        </xsl:for-each>
</xsl:template>

<!-- Display Title -->
<xsl:template  name="title" >

        <xsl:for-each select="/welcome/title" >

            <xsl:value-of select="."/>

        </xsl:for-each>
</xsl:template>

<!-- Diplay Page info_1 -->
<xsl:template  name="information1">

    <xsl:for-each select="welcome/info_1">

            <xsl:value-of select="."/>

    </xsl:for-each>
```

```
</xsl:template>

<!-- Diplay Page info_2 -->
<xsl:template  name="information2">

        <xsl:for-each select="welcome/info_2">

                <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>

<!-- Diplay Page info_3 -->
<xsl:template  name="information3">

        <xsl:for-each select="welcome/info_3">

                <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>


<!-- Link 1-->
<!--I'd rather create value strings that is the button names from the values held in the xml index file -->
<xsl:template  name="link1">
```

```
        <xsl:for-each select="/welcome/link_1" >

            <input type = "submit" value = "M-Login" onclick="check(2)" name="login"/>

        </xsl:for-each>
</xsl:template>

<!--Link 2-->
<xsl:template  name="link2">

        <xsl:for-each select="/welcome/link_2" >

            <input type = "submit" value = "M-Registration" onclick="check(3)" name="registration" />

        </xsl:for-each>

</xsl:template>

</xsl:stylesheet>
```

Login_HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:template match="/">

<html>

<head>
<title><xsl:call-template name="title_bar"/></title>

<script language="javascript">

        function check( type )
        {
                var flag=0;
                var name=document.request.input_user_id.value;
                var pass=document.request.input_password_id.value;


         if (flag==0)
          {
           if (name=="")
           {
            alert("User Name cannot be kept blank")
            flag=1;
            document.request.input_user_id.focus()
            }
          }
```

```
 if (flag==0)
 {
    if(pass=="")
    {
      alert("Password cannot be kept blank")
      flag=1;
      document.request.input_password_id.focus()
    }
 }

if (flag==0)
{

          var request =  type;

          document.request.requesttype.value = request;

          document.request.submit()
}



          <!--
          alert( "type is " + type );
          alert( "request is set to " + request )
          var request2 = document.request.requesttype.value;

          alert( "request type is set to " + request2 )
          -->
```

```
        }
</script>

</head>

<body>

<center>

<h1><b><xsl:call-template name="title"/></b></h1>

<br/>
<br/>

<img src="C:\apache\Tomcat 4.1\webapps\mmusic\images\m-music.gif" align="middle"/>

<br/>
<br/>

<h2><b><xsl:call-template name="information1"/></b></h2>

<table>
<form name="request" method="get" action="/mmusic/login">

        <tbody>
                <tr>
```

```
                <td><b><xsl:call-template name="input_id"/></b></td>
            </tr>

            <tr>
                <td><b><xsl:call-template name="input_password"/></b></td>
            </tr>

            <tr>
                <br/><br/><br/>
            </tr>

            <tr>


            <input type="hidden" name="requesttype" />
            <td><xsl:call-template name="link1"/></td>
            <td><xsl:call-template name="link2"/></td>



            </tr>

        </tbody>
        </form>
</table>

</center>

</body>
```

```
</html>

</xsl:template>

<xsl:template  name="title" >

        <xsl:for-each select="/login/title" >

            <xsl:value-of select="."/>

        </xsl:for-each>
</xsl:template>

<xsl:template  name="title_bar" >

        <xsl:for-each select="/login/title_bar" >

            <xsl:value-of select="."/>

        </xsl:for-each>
</xsl:template>

<xsl:template  name="information1" >

        <xsl:for-each select="/login/info_1" >

            <xsl:value-of select="."/>

        </xsl:for-each>
</xsl:template>
```

```
<xsl:template  name="link1">


        <xsl:for-each select="/login/link_1" >

            <input type = "submit" value = "M-Login" onclick="check( 2 )" />

            <!--onclick="check(2)" name="login"-->

        </xsl:for-each>
</xsl:template>

<!--Link 2-->
<xsl:template  name="link2">

        <xsl:for-each select="/login/link_2" >

            <input type = "submit" value = "M-Forgotten Password"/>
                <!--onclick="check(2)" name="forgotten" -->

        </xsl:for-each>

</xsl:template>

<xsl:template  name="input_id">

        <xsl:for-each select="/login/input_user_id" >

            <xsl:value-of select="."/>
```

```
                <xsl:text>                </xsl:text>
                <input type = "text">
                        <xsl:attribute name="name">
                                <xsl:value-of select="name()"/>
                        </xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>
<xsl:template  name="input_password">
        <xsl:for-each select="/login/input_password_id" >
                <xsl:value-of select="."/>
                <xsl:text>                </xsl:text>
                <input type="password">
                        <xsl:attribute name="name">
                                <xsl:value-of select="name()"/>
```

```
                    </xsl:attribute>

                </input>

            </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Registration_HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>
	<head>
		<title><xsl:call-template name="title_bar"/></title>

		<script language="javascript">

		function check( type )
		{
			var pass = document.request.input_password_id.value
			var repass = document.request.input_retype_id.value

			if( pass != repass )
			{
				alert( "password do not match")
				document.request.input_password_id.focus();
			}
			else
			{
				document.request.method = get;
			}
		}
		</script>
	</head>
```

```
        <body>

        <form action = "/mmusic/registration" name="request">

        <center>

        <h1><b><xsl:call-template name="title"/></b></h1>

        <br/>
        <br/>

        <img src="c:\images\m-music.gif" align="middle"/>

        <h2><b><xsl:call-template name="information1"/></b></h2>

        </center>


        <b><xsl:call-template name="input_user"/></b>

        <p>

        <b><xsl:call-template name="input_music"/></b>
        </p>

        <xsl:call-template name="submit"/>
        </form>
        </body>
</html>
```

```
</xsl:template>
<!--end calls-->
<xsl:template name="title">
        <xsl:for-each select="registration/title">
        <xsl:value-of select="."/>
        </xsl:for-each>
</xsl:template>
<xsl:template name="title_bar">
        <xsl:for-each select="registration/title_bar">
        <xsl:value-of select="."/>
        </xsl:for-each>
</xsl:template>
<xsl:template name="information1">
        <xsl:for-each select="registration/info_1">
        <xsl:value-of select="."/>
        </xsl:for-each>
```

```
</xsl:template>

<xsl:template name="input_user">

                Enter your personal details
                <br/>
        <xsl:for-each select="registration/input_user/*">
        <p>
        <xsl:choose>


                <!--Document this, the problem is solved, booooooooooooooowwwwwwwwwwww-->
                <xsl:when test='@type = "text"'><xsl:text>    </xsl:text>

                <xsl:value-of select="."/>
                <input type = "text">
                                <xsl:attribute name="name">
                        <xsl:value-of select="name()"/>
                </xsl:attribute>
          </input>
                <br/>

                </xsl:when>

                <xsl:when test='@type = "password"'>

                <xsl:value-of select="."/> <xsl:text>    </xsl:text>
                <input type = "password">
                                <xsl:attribute name="name">
                        <xsl:value-of select="name()"/>
```

```
                </xsl:attribute>
            </input>
                <br/>

                </xsl:when>

        </xsl:choose>

        </p>
        </xsl:for-each>

</xsl:template>

<xsl:template name="input_music">

        Enter your first music preference
        <br/>
        <xsl:for-each select="registration/input_music/*">

                <input type = "radio">
                        <xsl:attribute name="name">
                                <xsl:text>music_1</xsl:text>
                        </xsl:attribute>
                        <xsl:attribute name="value">
                                <xsl:value-of select="@name"/>
                        </xsl:attribute>
                </input><xsl:value-of select="."/><br/>

        </xsl:for-each>

        <br/>
```

```
	Enter your second music preference
	<br/>

	<xsl:for-each select="registration/input_music/*">

		<input type = "radio">
			<xsl:attribute name="name">
				<xsl:text>music_2</xsl:text>
			</xsl:attribute>
			<xsl:attribute name="value">
				<xsl:value-of select="@name"/>
			</xsl:attribute>
		</input><xsl:value-of select="."/><br/>

	</xsl:for-each>


</xsl:template>

<xsl:template name="submit">

<xsl:for-each select="registration/link_1">

	<input type = "submit">
		<xsl:attribute name="value">
			<xsl:value-of select="."/>
		</xsl:attribute>
	</input>

</xsl:for-each>
```

```
</xsl:template>

</xsl:stylesheet>
```

Music_profle_HTML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/">

<html>

<head>
<title><xsl:call-template name="title"/></title>
</head>

<body>
<!--<form action = "/mmusic/checker" method = "get">-->
<center>

<h1><b><xsl:call-template name="title"/></b></h1>

<br/>
<br/>

<img src="C:\apache\Tomcat 4.1\webapps\mmusic\images\m-music.gif"  align="middle"/>
</center>
<br/>
<br/>
<br/>
<br/>

<h2><b><xsl:call-template name="type"/> music defenition:</b></h2>
<br/>
```

```
<p>
<xsl:call-template name="defenition"/>
</p>
<br/>
<p>
<xsl:call-template name="general"/>
</p>
<h2><b><xsl:call-template name="type"/>music timeline:</b></h2>

<table>
	<tbody>
		<tr>
			<xsl:call-template name="timeline"/>
		</tr>
	</tbody>
</table>


</body>
</html>
</xsl:template>

<xsl:template name="title">

	<xsl:for-each select="music_profile/title">

	<xsl:value-of select="."/>

	</xsl:for-each>

</xsl:template>
```

```
<xsl:template name="type">
	<xsl:for-each select="music_profile">
		<xsl:value-of select="@type"/>
	</xsl:for-each>
</xsl:template>

<xsl:template name="defenition">
	<xsl:for-each select="music_profile/defenition/*">
		<xsl:value-of select="."/>
	</xsl:for-each>
	<!--<xsl:for-each select="music_profile/muisc_defenition/*">
	<xsl:value-of select="."/>
	</xsl:for-each>-->
</xsl:template>

<xsl:template name="general">
<xsl:for-each select="music_profile/general/*">
<xsl:value-of select="."/>
```

```
<!--<xsl:for-each select="music_profile/general_information/*">-->

<!--<value-of select="."/>-->
<!--</xsl:for-each>-->
</xsl:for-each>
</xsl:template>


<xsl:template  name="timeline">

          <xsl:for-each select="/music_profile/timeline">
          <xsl:for-each select="/music_profile/timeline/*">
          <tr>
         <td valign="top"><b><xsl:value-of select="@year"/></b></td>
         <td><xsl:value-of select="."/>
         <xsl:for-each select="/music_profile/timeline/*/*">

         <xsl:value-of select="."/>

         </xsl:for-each>
         </td>
         </tr>
          </xsl:for-each>
          </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Index_WML.xsl

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" indent="yes"
  doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
  doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"/>

<!--Begin templete call segment embedded in WML -->
<xsl:template match="/">



<wml>
        <!-- M-muisc inital page -->
        <!--Reported problems with the do tags, the type cannot be repeated so a combination of
                options and unknow along with accept is used-->

        <!--XSL with two cards, welcome and home -->

        <template>
                <do type="accept" label="M-Home">
                        <go href="#welcome"/>
                </do>
        </template>

        <!-- WELCOME PAGE -->
        <card id="welcome" title="M-Music">
                <onevent type="ontimer">
```

```
                <go href="#home"/>
        </onevent>
        <timer value="40"/>

        <p align="center">
                        <img src="m-music.wbmp" alt="M-muisc" align = "middle"/>
        </p>
</card>

<!-- HOME PAGE -->
<card id="home" title="M-Home">

        <do type="options" label="M-Login">
                <go href = "#login"/>
        </do>

        <do type="unknow" label="M-Registration">
                <go href = "#registration"/>
        </do>

        <p align="center">

        <big>
        <b><xsl:call-template name="information1"/></b>
        </big>

        <br/>
        <br/>

        <b><xsl:call-template name="information3"/></b>
```

```
                <br/>
                <br/>

                <b><xsl:call-template name="WAP_info"/></b>

                <br/>
                <br/>

                <b><xsl:call-template name="WAP_info_2"/></b>

                </p>

        </card>

</wml>

</xsl:template>
<!--End templete call-->



<!--Begin template definition -->

<!-- Diplay Page info_1 -->
<xsl:template  name="information1">

        <xsl:for-each select="welcome/info_1">

                <xsl:value-of select="."/>

        </xsl:for-each>
```

```
</xsl:template>


<!-- Diplay Page info_3 -->
<xsl:template  name="information3">

        <xsl:for-each select="welcome/info_3">

                <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>


<!-- Diplay Page WAP_info -->
<xsl:template  name="WAP_info">

        <xsl:for-each select="welcome/wap_info">

                <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>

<!-- Diplay Page WAP_info -->
<xsl:template  name="WAP_info_2">

        <xsl:for-each select="welcome/wap_info_2">
```

```
                <xsl:value-of select="."/>
            </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Login_wml.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" indent="yes"
  doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
  doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"/>

<xsl:template match="/">

<!--Post tags required-->
<!--!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Links to other cards in other files required
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!11-->

<wml>
	<template>
		<do type="accept" label="M-Home">
			<go href="#welcome"/>
		</do>
	</template>


	<!-- LOGIN PAGE -->
	<card id="login" title="M-Login">
		<do type="options" label="M-Registration">
			<go href="#registration"/>
		</do>
		<do type="unknow" label="M-Submit">
			<go href="#profile"/>
		</do>
```

```
                <do type="accept" label="M-Forgotten Password">
                        <go href="#forgotten"/>
                </do>

                <p align="center">
                        <big>
                                <b><xsl:call-template name="wap_info"/> </b>
                        </big>
                        <br/>
                        <br/>
                        <xsl:call-template name="input_id"/>
                        <!--<input name="user_id" title="input user id"/>-->
                        <br/>
                        <xsl:call-template name="input_password"/>
                        <!--<input name="password" type="password" title="input password"/>-->
                </p>

        </card>
</wml>

</xsl:template>

<xsl:template name="wap_info">

        <xsl:for-each select="login/wap_info">

                        <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>
```

```
<xsl:template  name="input_id">
        <xsl:for-each select="/login/input_user_id" >
            <xsl:value-of select="."/>
            <input title="Input user id">
                <xsl:attribute name="name">
                    <xsl:value-of select="name()"/>
                </xsl:attribute>
            </input>

        </xsl:for-each>
</xsl:template>

<xsl:template  name="input_password">
        <xsl:for-each select="/login/input_password_id" >
            <xsl:value-of select="."/>
            <input title="Input password" type="password">
```

```
                <xsl:attribute name="name">
                        <xsl:value-of select="name()"/>
                    </xsl:attribute>
                </input>
            </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Registration_WML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" indent="yes"
  doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
  doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"/>

<!--Begin templete call segment embedded in WML -->
<xsl:template match="/">



<wml>
        <!-- M-muisc inital page -->
        <!--Reported problems with the do tags, the type cannot be repeated so a combination of
                options and unknow along with accept is used-->

        <!--XSL with two cards, welcome and home -->
                <template>
                <do type="accept" label="M-Home">
                        <go href="#welcome"/>
                </do>
        </template>

        <!-- REGISTRATION -->
        <card id="registration" title="M-registration">

                <do type="options" label="M-Login">
                        <go href="#login"/>
```

```
                </do><!--needs a templete-->

                <do type="unknow" label="M-Submit">

                        <go href="/mmusic/init_functions.wmls#cal( '$input_password', '$input_retype' )"/>

                        <!--<go href = "init_functions.wmls#cal( 2, 3 )"/>-->
                        <!--<go href="#profile"/>-->
                </do>

                <p align="center">
                        <big>
                                <b><xsl:call-template name="title"/></b>
                        </big>

                        <br/>
                        <br/>

                        <xsl:call-template name="input_user"/>

                        <xsl:call-template name="input_music"/>

                </p>
        </card>
</wml>
</xsl:template>

<xsl:template name="title">

        <xsl:for-each select="registration/title">
```

```
        <xsl:value-of select="."/>

        </xsl:for-each>

</xsl:template>

<xsl:template name="input_user">

        Enter your personal details
        <br/>
        <br/>

        <xsl:for-each select="registration/input_user/*">

        <xsl:choose>

                <xsl:when test='@type = "text"'>

                <xsl:value-of select="."/>

                        <input>
                                <xsl:attribute name="name">
                                <xsl:value-of select="name()"/>
                                </xsl:attribute>
                                <xsl:attribute name="title">
                                <xsl:value-of select="."/>
                                </xsl:attribute>
                        </input>
                        <br/>
                </xsl:when>
```

```
            <xsl:when test='@type = "password"'>

            <xsl:value-of select="."/>

                  <input type="password">
                        <xsl:attribute name="name">
                        <xsl:value-of select="name()"/>
                        </xsl:attribute>
                        <xsl:attribute name="title">
                        <xsl:value-of select="."/>
                        </xsl:attribute>

                        <xsl:attribute name="size">
                        <xsl:value-of select="@size"/>
                        </xsl:attribute>

                        <xsl:attribute name="maxlength">
                        <xsl:value-of select="@maxlength"/>
                        </xsl:attribute>

                  </input>
                  <br/>

            </xsl:when>

      </xsl:choose>
      </xsl:for-each>

</xsl:template>
```

```
<xsl:template name="input_music">

        Enter your music preference
        <br/>

        <select name="music_type" title="Music" multiple="true">

        <xsl:for-each select="registration/input_music/*">

        <option>
        <xsl:attribute name="value">
        <xsl:value-of select="@name"/>
        </xsl:attribute>
        <xsl:value-of select="."/>
        </option>

        </xsl:for-each>

        </select>



</xsl:template>


</xsl:stylesheet>
```

Music_Profile_WML.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" indent="yes"
  doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
  doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"/>

<!--Begin templete call segment embedded in WML -->
<xsl:template match="/">



<wml>
        <!-- M-muisc inital page -->
        <!--Reported problems with the do tags, the type cannot be repeated so a combination of
                options and unknow along with accept is used-->

        <!--XSL with two cards, welcome and home -->
                <template>
                <do type="accept" label="M-Home">
                        <go href="#welcome"/>
                </do>
        </template>

<card id="profile" title="M-Music">
        <!--<xsl:call-template name="card_title"/>-->
        <!--<card id="registration" title="M-registration">-->
                <!-- create link for profile-->
                <do type="options" label="M-Profile">
```

```
                <go href="#profile"/>
            </do>

            <p align="center">
                <b><xsl:call-template name="title"/></b>
                <br/>
                <br/>
                <b><xsl:call-template name="type"/> Music defenition</b>
                <br/>
                <br/>
                <xsl:apply-templates select="music_profile/defenition/muisc_defenition"/>

                <br/>
                <br/>
                <xsl:apply-templates select="music_profile/general/general_information"/>
                <br/>
                <br/>
                <b><xsl:call-template name="type"/> Music timeline</b>
                <br/>
                <br/>

                <xsl:call-template name="timeline"/>



            </p>
        </card>
</wml>
</xsl:template>

<xsl:template name="music_defenition">
```

```
                        <xsl:for-each select="music_profile/muisc_defenition">
                        <xsl:choose>
                        <xsl:when test= ' @type = "wap" '>
                                <xsl:value-of select="."/>
                        </xsl:when>
                        <xsl:when test='@type = "html" '>
                        NO tetx
                        </xsl:when>
                        </xsl:choose>
                        </xsl:for-each>

                        <!--<xsl:call-template name="build"/>-->

                        <!--</card>-->
</xsl:template>

<!--
<xsl:template name="card_title">

<xsl:for-each select="music_profile">
<card>
        <xsl:attribute name="id">
        <xsl:value-of select="@wap_type"/>
        </xsl:attribute>
        <xsl:attribute name="title">
        <xsl:text>M-</xsl:text><xsl:value-of select="@type"/>
        </xsl:attribute>

</xsl:for-each>
```

```
</xsl:template>
-->


<xsl:template name="title">

	<xsl:for-each select="music_profile/title">

	<xsl:value-of select="."/>

	</xsl:for-each>

</xsl:template>

<xsl:template name="type">

	<xsl:for-each select="music_profile">

		<xsl:value-of select="@type"/>

	</xsl:for-each>
</xsl:template>

<xsl:template match="muisc_defenition">

	<!--<xsl:for-each select="music_profile/muisc_defenition">-->

		<xsl:value-of select="."/>

	<!--</xsl:for-each>-->
```

```
</xsl:template>

<xsl:template name="general">

<xsl:for-each select="music_profile/general_information">

<xsl:value-of select="."/>

</xsl:for-each>
</xsl:template>


<xsl:template  name="timeline">

        <xsl:for-each select="music_profile/timeline/line">
        <b><xsl:value-of select="@year"/></b>
        <br/>
        <xsl:value-of select="."/>
        <br/><br/>


        </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```