I-Shop
A Context-Aware Cross-Platform Shopping Advisor

by

Ishita Jain
BCA, Devi Ahilya University, 2009

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Ishita Jain, 2013
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy
or other means, without the permission of the author.
Supervisory Committee

I-Shop
A Context-Aware Cross-Platform Shopping Advisor

by

Ishita Jain
BCA, Devi Ahilya University, 2009

Supervisory Committee

Dr. Hausi A. Müller, Department of Computer Science
Supervisor

Dr. Alex Thomo, Department of Computer Science
Departmental Member
Abstract

Supervisory Committee

Dr. Hausi A. Müller, Department of Computer Science
Supervisor

Dr. Alex Thomo, Department of Computer Science
Departmental Member

This thesis presents the design and implementation of I-Shop, a context-aware, shopping smartphone application designed to provide shoppers with relevant advertisements for product and services available in close proximity. We argue that current context-aware mobile applications exhibit significant limitations in the following domains: (1) use of context, (2) invasion of privacy, (3) spam management, and (4) platform dependency. The proposed context model attempts to tackle these shortcomings by exploiting available contextual information from social media networks such as Facebook. Our goal is to use a user’s personal information, such as their native language and personal interests, to direct the most relevant advertisements to them. To alleviate any privacy issues, a user’s personal information is never sent out to any back-end services and only apply the filters locally. In addition, unlike most other predictive approaches that track the user’s location history, we follow a reactive approach which triggers only when the user is close to a shopping area. When a user arrives to a particular shopping area, the application asks whether she wishes to view any advertisements of local products and services. Upon approval, the application retrieves deals on products including services sorted by domain from databases, such as Groupon and our custom
extended deals database. Finally, the application filters the retrieved data according to personal interests and then displays the results.

As a proof of concept, we designed and implemented the I-Shop prototype application. We built I-Shop as a hybrid application using IBM’s state-of-the-art Worklight infrastructure. This approach lets developers optimize their time and effort; enabling a “write once, deploy everywhere” development model that not only reduces development costs but also increases application performance by providing a combination of native and web capabilities. In addition, I-Shop also leverages several features offered by the IBM Worklight infrastructure including cross-platform support, direct update, internalization, and integration of third-party libraries and toolkits.
# Table of Contents

Supervisory Committee ..................................................................................................................... ii  
Abstract .............................................................................................................................................. iii  
Table of Contents ................................................................................................................................. v  
List of Tables .......................................................................................................................................... viii  
List of Figures ......................................................................................................................................... ix  
Acknowledgments ................................................................................................................................. xi  
Chapter 1 Introduction .......................................................................................................................... 1  
  1.1 Motivation ........................................................................................................................................ 1  
  1.2 Problem Statement ......................................................................................................................... 2  
  1.3 Approach ....................................................................................................................................... 4  
  1.4 Contributions ................................................................................................................................. 5  
  1.5 Thesis Outline ............................................................................................................................... 5  
Chapter 2 Background ........................................................................................................................... 6  
  2.1 Context and Context-Aware Systems ......................................................................................... 6  
    2.1.1 Context .................................................................................................................................... 6  
    2.1.2 Context-Aware Applications ................................................................................................. 8  
  2.2 Location Based Services ............................................................................................................. 9  
    2.2.1 Architecture of Location Based Services ............................................................................. 9  
    2.2.2 Use of Location Based Services ......................................................................................... 11  
  2.3 Location Based Advertising ....................................................................................................... 15  
    2.3.1 Spam ...................................................................................................................................... 16  
    2.3.2 Privacy ................................................................................................................................... 16  
  2.4 IBM Worklight ............................................................................................................................ 17  
    2.4.1 Overview ............................................................................................................................... 17  
    2.4.2 Studio ..................................................................................................................................... 19  
    2.4.3 Server .................................................................................................................................... 26  
    2.4.4 Device Runtime .................................................................................................................... 27  
    2.4.5 Console .................................................................................................................................. 28
2.4.6 AppCenter ................................................................. 28
2.4.7 Limitations ................................................................. 28
2.5 Summary .................................................................. 29

Chapter 3 I-Shop: A Context-Aware Shopping Advisor .................. 30
3.1 Key Features of I-Shop ................................................. 30
  3.1.1 Lightweight Client Application .................................. 30
  3.1.2 Proactive and Reactive Approaches to User Recommendations ................................................. 31
  3.1.3 Cross-Platform Support ............................................. 32
  3.1.4 Personalization .......................................................... 33
  3.1.5 Content Generation .................................................. 34
  3.1.6 Statelessness ........................................................... 34
  3.1.7 Self-Updating .......................................................... 35
  3.1.8 Internationalization .................................................. 36
  3.1.9 Hybrid Coding ........................................................ 36
  3.1.9 Scalability ............................................................... 38
  3.1.10 Dynamic Control .................................................... 40
  3.2 Scenario Based Interaction Model for I-Shop ....................... 41
3.3 Summary .................................................................. 46

Chapter 4 I-Shop Data Sources and Application Model .................. 47
4.1 Data Sources of I-Shop ................................................... 47
  4.1.1 Groupon ................................................................. 47
  4.1.2 I-Shop and the Groupon Deals API .............................. 48
  4.1.3 Facebook ............................................................... 50
4.2 I-Shop: Application Model .............................................. 58
  I-Shop: Client ................................................................. 59
  I-Shop: Server ............................................................... 71
4.3 Summary .................................................................. 74

Chapter 5 I-Shop Implementation and Evaluation ......................... 75
5.1 The I-Shop Implementation Model ...................................... 75
5.2 Evaluation of I-Shop ...................................................... 76
  5.2.1 Overview of the Experiment .......................... 76
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2 Evaluation by Participant 1</td>
<td>77</td>
</tr>
<tr>
<td>5.2.3 Evaluation by Participant 2</td>
<td>80</td>
</tr>
<tr>
<td>5.3 Summary</td>
<td>82</td>
</tr>
<tr>
<td>Chapter 6 Conclusions</td>
<td>83</td>
</tr>
<tr>
<td>6.1 Summary</td>
<td>83</td>
</tr>
<tr>
<td>6.2 Contributions</td>
<td>85</td>
</tr>
<tr>
<td>6.3 Future Directions</td>
<td>85</td>
</tr>
<tr>
<td>6.3.1 Areas for Improvement</td>
<td>85</td>
</tr>
<tr>
<td>6.3.2 Public Release Potential</td>
<td>86</td>
</tr>
<tr>
<td>6.3.3 Commercial Potential</td>
<td>87</td>
</tr>
<tr>
<td>Bibliography</td>
<td>88</td>
</tr>
<tr>
<td>Appendix A Glossary</td>
<td>94</td>
</tr>
</tbody>
</table>
List of Tables

Table 1: App development across different mobile platforms........................................ 21
Table 2: Results based on user’s wishlist (cosmetics) for Participant 1.......................... 78
Table 3: Results based on user’s selection (shoes) for Participant 1............................... 79
Table 4: Results based on user’s wishlist (sunglasses) for Participant 2 ....................... 81
Table 5: Results based on user’s selection (home and décor) for Participant 2............... 81
Table 6: Distribution cost across different mobile platforms ........................................ 86
List of Figures

Figure 1: Classification of context [17] ........................................................................................................ 7
Figure 2: Pull-Based Model [23] .................................................................................................................. 10
Figure 3: Poll-Based Model [23] ................................................................................................................ 11
Figure 4: Push Based Model [23] ................................................................................................................. 11
Figure 5: Overview of IBM Worklight components [12] ............................................................................. 19
Figure 6: File Structure of IBM Worklight applications ........................................................................... 20
Figure 7: Overview of mobile app building approaches [12] ................................................................... 23
Figure 8: IBM Worklight’s Shell approach [12] ......................................................................................... 25
Figure 9: Dialog box for Wishlist creation in I-Shop ............................................................................... 31
Figure 10: Adding different environment to a single application ............................................................ 33
Figure 11: Custom I-Shop logos for female and male users ..................................................................... 34
Figure 12: IBM Worklight direct update architecture [12] .................................................................... 35
Figure 13: I-Shop’s file system structure .................................................................................................. 37
Figure 14: Hybrid app: Interaction between web and native components [58] ..................................... 38
Figure 15: Adapter paradigm adopted by the IBM Worklight platform [12] ........................................ 39
Figure 16: Groupon HTTP adapter for I-Shop ......................................................................................... 40
Figure 17: IBM Worklight console for I-Shop ........................................................................................... 41
Figure 18: Facebook dialog box for logging into I-shop .......................................................................... 42
Figure 19: I-Shop’s custom user interface ................................................................................................. 42
Figure 20: Internationalization query ...................................................................................................... 42
Figure 21: I-Shop’s scenario based interaction model ............................................................................. 43
Figure 22: I-Shop Wishlist ......................................................................................................................... 44
Figure 23: I-Shop item selection ............................................................................................................... 44
Figure 24: I-Shop user interface displaying deals .................................................................................. 46
Figure 25: Sample deal offered ................................................................................................................ 46
Figure 26: Screenshot of a sample deal on www.groupon.com illustrating the values used by I-Shop .... 48
Figure 27: getGrouponDeals response in JSON format ....................................................................... 49
Acknowledgments

This research project would not have been possible without the support of many people. I would like to thank my supervisor, Dr. Hausi A Müller, who provided assistance, moral support and guidance throughout my studies at University of Victoria. I also want to express my gratitude to Dr. Alex Thono for providing feedback on my thesis.

I am indebted to all the members of the Rigi research group for their valuable feedback and being great friends. In particular, I would like to acknowledge Ron Desmarais who played a significant role in devising and implementing the ideas expressed in this thesis.

Last but not the least, I am very grateful to my family and cousins for their immense support and love throughout this entire journey.
Chapter 1 Introduction

This chapter introduces the reader to the work described in this thesis. Recent developments in web technologies, mobile platforms, and context-aware applications form the basis and motivation of this thesis. The problem statement argues and explains that current location based smartphone and tablet applications generate significant trust and frustration issues among end users. The approach description presents a solution that aims to improve the user experience with respect to this problem. The chapter concludes with a roadmap and the contributions of this thesis.

1.1 Motivation
Rapid advancement in information technology has changed our day-to-day lives tremendously over the last quarter century. It has played a key role in globalization and added pace and stress to our lives. The emergence of web technology has given rise to many breathtaking innovations. Today, these advancements make it possible to obtain information anywhere using mobile devices, such as smartphones and tablets. Advancements in technology, especially in the fields of software-intensive systems and smart applications, have completely transformed our assumptions of progress over the past decade. With the realization of context in computing, these systems are becoming increasingly intelligent, by recognizing and exploiting human behavior and other contextual parameters to drive their actions and produce more relevant results.

Context-aware computing has become a focus for a significant part of the software engineering research community. Context can be defined as “any information that can be used to characterize the situation of an entity” [1]. Context can be anything: physical location, time of day, season, local weather, web browsing history, as well as personal information and interests. Context can be leveraged by software applications to provide intelligent recommendations to a user seeking information. In part due to the ubiquitous location and time information provided by GPS (global positioning system) today location and time are two of the most important context parameters and prove to be critical components in many domains including transportation and logistics, medicine,
customer relationships, management, advertising, and marketing. Thus, location and time are relevant in many different aspects of decision making.

Location based advertising (LBA) is a new form of mobile marketing. It uses different types of location tracking features built into mobile devices to derive context to drive the most relevant advertisements to consumer. As stated by Bruner and Kumar “LBA refers to marketer-controlled information specially tailored for the place where users access an advertising medium” [9]. Location based technology has tremendous potential to win out against other types of marketing as the primary mode of delivering targeted advertisements to mobile devices.

1.2 Problem Statement
Because of the ubiquitous nature of smartphones, mobile advertising has become very popular in the advertising and marketing industry. Advertising on mobile applications usually leverages location to provide users with information on various products and services available within certain proximity range. Sometimes, it involves storing a user’s personal information at a third party (e.g., Amazon) to provide personalized recommendations, offers, and deals. These personalized advertisements are clearly more user-centric and more influential but still there is plenty of potential for improvement as outlined below:

Invasion of privacy
Context-aware computing including LBA often involves tracking people’s location [1]. Smartphones are extremely smart and truly personal tools. Darling states that “The fact that a mobile device is so personal can be both strength and a weakness” [2]. On the one hand, marketers can entertain, provide relevant information, build brand awareness, create loyalty, and influence purchasing decisions among their targeted customers through LBA. On the other hand, consumers can also get offended if their personal information, such as their location and activities, are being tracked by third parties. Halper said that a consumer’s privacy is a great concern [3]. It is crucial to understand their concerns of privacy and develop systems that could give them personalized information without entering into their private sphere.
Perception of spam
Another way of promoting businesses is through position aware services that rely on the device’s knowledge of its own location. These kinds of services do not track down their customers but instead react to their location. While this approach solves one issue, it spams devices with advertisements even when users are not interested. As a result, users perceive such services as spam and tend to run away from these services. It is important to filter the content targeted to their devices not only based on their location but also other contextual parameters that may influence their choices.

Limited use of context
Irrelevant or incomplete content generation is often frustrating for users when they need to access information quickly and accurately. As an example, consider a bald man who receives a coupon from a hair salon just because he is in close proximity of that service. Clearly, a bald man will not be attracted to this service; instead there is a good chance that he will be offended by such a coupon. Thus, it is critical to tailor the information to the user’s properties, needs, capabilities and interests that will most likely to influence their decisions. Besides that, because of the limitations of handheld portable devices, including small screen and limited resources, it is important to understand and leverage context related to devices for optimal utilization of resources. This can be accomplished by exploiting the context related to the person along with her location. For example, their interests, hobbies, current weather, location and time can all be used to filter recommendations to make them more relevant and personal.

Platform dependency
Today mobile device providers are in a fierce competition to sell their products. Some of the leading mobile device platforms include Android, iOS, Blackberry, Windows, and Symbian. Native mobile applications running on these platforms are usually built exclusively to run on only one of these platforms. On the one hand, building and maintaining multiple applications for all of these platforms is not only expensive but also time consuming as it requires unique technical expertise. On the other hand, web application built for mobile devices are not as powerful as native applications as they do not have access to many of the device specific components such as native widgets and
context. Due to this limitation, most of the application models that provide location based services are limited to the operating system that the device is running on. This thesis employs an application model that bridges the gap between these two types of applications. We use IBM’s Worklight framework for building hybrid applications that can be provisioned onto several platforms. IBM Worklight provides an open, comprehensive, and advanced mobile application platform for smartphones and tablets, helping organizations of all sizes to develop, connect, run, and manage HTML5 web applications and native applications efficiently as hybrid applications [4]. Thus, this application model leads to hybrid mobile applications that can be embedded into a native shell of the native mobile operating systems. IBM Worklight supports Android, iOS, Blackberry, and Windows platforms. The aim of this thesis is to develop an application model that addresses the aforementioned shortcomings of current location based advertising applications.

1.3 Approach
To alleviate the aforementioned design issues of location based advertising applications, we propose a generic application model for context-aware location based advertising applications that particularly addresses the following features:

1) Use a reactive approach to leverage a user’s location without keeping record of her visits to provide her with relevant information on demand.
2) Identify sources for context acquisition and demonstrate how it applies to commercial applications.
3) Build a cross-platform hybrid application model leveraging the IBM Worklight framework including standard technologies, an integrated development environment, mobile-optimized middleware, as well as a management and analytics console.
4) Incorporate the IBM Worklight adapter framework to provide scalability and modularity to facilitate application extensibility in the future.
5) Leverage the information available social media networks frequented by the user to acquire user preferences and personal information to enhance user experience, and personalization without intruding on her personal comfort zone.
We considered these challenges and features during the design and implementation of our I-Shop prototype application for smartphones. The resulting application is a context-aware, stateless shopping advisor that reacts to the user location taking user context and preferences into the account when delivering recommendations.

1.4 Contributions
The main contribution of this thesis is the application model for creating dynamic, stateless, portable, and context-aware smart applications for mobile devices. The implementation prototype, called I-Shop, demonstrates that our model is feasible and practical. In the implementation we leveraged the IBM’s state-of-the-art Worklight cross platform infrastructure to further enhance the usability, scalability and ubiquity of I-Shop. I-Shop uses core web technologies along with native capabilities to provide a rich set of functionalities and excellent performance.

1.5 Thesis Outline
Chapter 2 presents background information as foundation for the work presented in this thesis. It sheds light on the latest trends in mobile computing including context management and context-aware applications, in particular location based services (LBS), and IBM’s revolutionary Worklight product. Chapter 3 provides an overview of our prototype applications: I-Shop including its key features and functionalities. Chapter 4 presents the proposed generic application model. Chapter 5 discusses the implementation of I-Shop in detail. Chapter 6 draws some conclusions and presents directions for future work.
Chapter 2 Background

This chapter introduces the reader to the most important concepts of this thesis and the infrastructure used in our project. The discussion begins with an overview of context and context-aware systems. The discussion shifts to one of the applications of location based services: *location based advertising*, which motivated this research. We then concentrate on IBM Worklight and its infrastructure for developing mobile applications.

2.1 Context and Context-Aware Systems

With rapid advancements in technology, ubiquitous computing has become extremely influential. Ubiquitous computing refers to a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday life. Smartphones, wearable computers, autonomous vehicle control systems, and the previously-simple television are all examples of products that are designed to improve convenience and quality of life. However, there exists immense potential to make these systems even more intelligent by giving them capabilities to understand and interpret their environment or context and thereby autonomously adapt to current requirements.

2.1.1 Context

All living beings survive by learning to adapt to their surroundings. Using available sensory means, we build the most complete picture available of our current situation. Information regarding our current environment (or context) is automatically filtered, and relevant information is factored into our decision-making processes. These factors can be many different things from the emotional state of other human beings, to current climate conditions, or the time of day. Basically, any factor which can affect the behavior of an entity represents context.

The need for context in software engineering has increased in importance over the last few years, spurred by the massive growth in ubiquitous computing. Rich context data can be utilized both to augment a user’s input ability when interacting with traditional software, and also in situations where context changes rapidly. Significant research has
been focused on leveraging context to optimize smart applications. Dey, one of the great researchers in the field of context, defines context as follows [14].

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.”

In other words, context can be any information used to optimize the behavior of a system to improve its user experience. Contextual information can be made available to a system that uses it to improve human-computer or environment-computer interaction. Users in the age of ubiquitous computing have high expectations; they should be able to quickly, easily and comfortably access whatever information or services they need. With a wide variety of mobile devices used by a plethora of unique users, interesting new problems arise, and the need for context increases. Users need to obtain tailored information using the same services in dissimilar situations. Context can be used to help determine what information should be made available to the user [13].

Villegas et al. classified context to control and govern information in an adaptive environment as depicted in Figure 1.

![Figure 1: Classification of context](image)

Clearly, there are a wide variety of contexts available to leverage rich information and create more adaptive and dynamic systems.
2.1.2 Context-Aware Applications

“Applications that use context, whether on a desktop or on a mobile device in a ubiquitous computing environment, are called context-aware applications” [13]

Context-aware applications are concerned with the following challenges:

Acquisition of Context

It is important to understand the contextual parameters that can affect a system’s behavior. Acquiring every possible piece of contextual information is impossible due to limited resources and the limitations of modern hardware and sensors in particular. The best approach is to create a model representing the contextual parameters most critical to the subject application. This contextual information can be gathered from a wide variety of sources:

- Sensors: We are surrounded by numerous sensors. The modern smartphone is an excellent example of a source of sensors. GPS receivers, cameras, gyroscopes, compasses and so on are all commonly included. The sensors equipped in most mobile devices can efficiently accumulate dynamic contextual information such as a user’s location, direction of travel, or nearby audio sources. More specialized sensors are embedded into devices for touch, temperature, and air pressure measurement [14].

- Web: The Internet is an excellent source for contextual information. There are many web services available that provide instant information on topics such as weather, road conditions, personal interests, and activities related to social network applications such as Facebook or Twitter.

- User: The most accurate source of context is through the user input. Their preferences, statuses, interests, and requirements can be used to understand their current situations better.

Interpretation of Context

Once the contextual data is acquired, the application needs to select the relevant information. Researchers have devised many approaches and models to derive the most relevant information for a given system. For example, IBM’s ACRA (Autonomic Computing Reference Architecture) provides an abstract model using feedback loops to
work on individual contexts at different levels of abstraction. It adopts a sense-plan-act approach to tailor the behavior of the system based on context acquired in the sensing phase and policies defined in the planning phase.

**Manipulating application behavior**
System changes are implemented in accordance with the resulting contextual information.

### 2.2 Location Based Services
Among most context-aware services, *location based services (LBS)* are the most popular. They provide integration of geographic location and software services [18]. The incorporation of location and navigation technology, such as the Global Positioning System (GPS) and other factors including cell tower triangulation and nearby WiFi networks into mobile phones, has created a new class of location-aware devices. These allow users to communicate with others and sometimes even determine the location of others using their own device. These services aim to offer personalized mobile transactions for individuals in specific locations at specific times. Among all contextual parameters, location has moved to the forefront due to the critical nature of location to many businesses and service providers.

Some of the most popular commercial location based services, such as Facebook, allow a user to *check in* at a location, which lets her friends know where she is. The location based service that this thesis concentrates on informs a user of relevant deals and offers regarding nearby products and services by leveraging information gained from their mobile device and social networking profiles.

#### 2.2.1 Architecture of Location Based Services
Based on the flow of information, Kumar, Qadeer and Gupta identified and defined the following basic architectures for location based services [23].
Pull-Based Model
A location proxy is between the client and the LBS application as depicted in Figure 2. When the client initiates a request to the application, her location information is attached to the request by the location proxy and then forwarded to the LBS application. In this case, the location aware service delivers the requested content when the client pulls the information from these services. Usually, the location proxy is integrated into a middleware infrastructure upon which a service, such as IBM WebSphere Everyplace Suite, is deployed. This model allows a developer to build and deploy an LBS-enabled application and convert existing applications to LBS-enabled applications easily, with the added benefit of making location inquiries transparent to the user.

Poll-Based Model
The LBS application actively sends location requests via a well-defined or industry standard location interface to the location server (LS), which is responsible for determining the location of the requested client. In this model, the LBS application keeps polling the LS or queries the LS on demand to answer questions from the client. The advantage of this model, as depicted in Figure 3 is that more advanced location functionalities, such as a periodic location report supported by WAP, could be supported, and a standard location interface makes widely distributed location aware computing (e.g., location Web Service) possible.
Push-Based Model
The LBS application pushes location-aware information to the client according to user preferences by tracking the position of different mobile users. The push model, depicted in Figure 4 opens the window to delivering correct information to the right people at the right time and location. The drawback of this model centers on privacy concerns. The fact that their location is being constantly monitored may irritate the user.

2.2.2 Use of Location Based Services
Location based services use geographic position to provide smarter information and entertainment services. Services provided through this type of technology span a broad range of purposes. In fact, this technology is frequently used to enhance emergency, personal, and professional services in many ways. Some of these services are described below:

Public Safety and Emergency Services
- The first application of this technology is in the emergency services industry to quickly determine the location of emergency callers [38]. This has proven such a valuable service that governments in Canada, USA, Europe, and Japan have approved legislation requiring mobile network operators to provide location data to emergency services.
• Location aware services have been used in areas prone to natural disasters around the globe. When there is a danger of a tsunami or earthquakes in particular area, the local residents can be warned using SMS and Cellular Broadcasting Services (CBS) well in advance [38].

• To reach every nook and cranny in case of an emergency, emergency personnel need to determine exact locations to deploy hospitals, fire stations, and police stations. These decision makers have greatly benefitted from this technology.

Consumer Services
Today, location based services have become an integral part of human beings in our society. Consumers use these services many times each day whether they know it or not. Location is an integral part of everything we do and it is used by many services to provide contextual information in areas such as:

• Navigation: Devices such as in-car GPS systems and mobile map applications use LBS to provide navigation assistance to their users, often complemented with traffic data to help users select their optimal route.

• Location based advertising: Advertisements of discounts and offers from businesses are sent to potential customers based on proximity and other contextual information that can influence shopping behavior.

• Location based search: There exist many applications, such as Yelp, that allow a user to access local services or find detailed information about services and businesses in a particular area. For example, a listing of show times and ratings for movies being played in a particular city.

• Location based mobile gaming: Uses the position of the user’s handset to deliver augmented gameplay experiences and contents.

• Location based reminders: Users can now use such services to create to-do lists to alert the user when she passes within close proximity of a location she need to be aware of or visit to complete her task.
Enterprise Services
Location based business intelligence has transformed the way the enterprise sector used to function. Whether in retail or marketing, distribution, or customer relationship management, location has become an integral part of each domain in the enterprise sector. Location based services enable all business users to gain critical insights by combining spatial data with other business data. Some of the most common implications are discussed below.

LBS in Retail
Retail chains, whether expanding or contracting, planning inventory production, distribution, or marketing, use location intelligence at every step by uncovering dependency relationships between stores, products, and demographics that affect sales. For example:

- Select sites for new outlets or warehouses based on the geographic distribution of their most profitable customers, the proximity of competing stores, and transportation routes.
- Analyze sales of different products and services across geographically distributed locations.
- Maintain optimal inventory across stores and distribution channels.
- Forecast sales patterns for new products based on local demographics.

LBS in Marketing and Advertising
As mobile technology continues to grow in popularity and sophistication, so do location based advertising and marketing strategies. Location information can be used to improve advertising and marketing in many ways:

- Enable better understanding of local demographics to devise advertising campaigns that target local populations.
- Help determine the effectiveness of advertising campaigns and then, based on the level of success, look for similar geographical patterns to try the same advertising strategies.
• Through mobile advertising, a user can be provided with ads based on their proximity to a business.

• Public transportation vehicles are now equipped with digital screens to display advertisements based on the current destination.

LBS in Transportation and Logistics
Location is crucial in creating optimal routes in transportation. Business officials can leverage location based services augmented with map technology to:
• Understand consumption across different locations and distribute their product accordingly to avoid over- or under-stocking inventory.

• Analyze various factors such as traffic, weather, and road conditions to estimate the best transportation route across various distribution channels and minimize cost.

LBS in Education and Research
Spatial information and capabilities provided by location based services are actively used in schools, universities, the research community, libraries, and museums.

• Libraries and museums use Geographic Information System- (GIS) powered information tools to educate students and visitors about connections between places, people, and historic events. [39] Ahmad et al and Sisgorgeo explain many such scenarios where breathtaking discoveries were made using location technology [40].

• Spatial information helps researchers and statisticians investigate trends and relationships. For instance, scientists at a South Dakota research center are remapping the location of the bones of Ice Age mammoths with GIS to preserve the context of the site for further analysis and research [42].

LBS in Telecommunications
Location intelligence (LI) is a key factor in the decision making processes of telecommunications companies as well. Executives use LI to roll out wireless networks, manage assets, and market their services effectively.
To provide the most efficient and effective network coverage to customers, network managers need to determine the optimal location to deploy a cellular tower. They then need to acquire rights from property owners, and ensure that local, regional, and federal land use and zoning regulations are taken into account.

Marketers and advertisers can use location information to tie existing Business Intelligence (BI) and GIS data to demographic data and allocate advertising capital accordingly, making it possible to focus their efforts in areas where they will have the best return on investment [38].

2.3 Location Based Advertising

Over the years, location based advertising has grown to become the most popular medium for advertising. Advertisers can now promote their products and services to people who have a higher chance of becoming customers. While not strictly limited, a potential customer’s mobile device is the main target for this kind of advertising strategy because of their ubiquitous nature and how they can provide not only geographical but also contextual information and allow users to demonstrate preferences. This rich information can be exploited by an advertiser to promote their business using targeted marketing.

To illustrate such a scenario, consider a potential customer “Kayla” standing in front of an Indian restaurant, using a smartphone application that incorporates location based advertising. The application makes it possible for an advertiser to know that Kayla is standing in front of an Indian restaurant and sends her relevant ads instantly. These ads can be in the form of a coupon or just information regarding enticing Indian dishes. Perhaps the deal is enticing for Kayla, and since it is currently being offered within close proximity, she might choose that restaurant and redeem the coupon.

This form of advertising has been shown to be effective [29] as it offers an incentive to the customer that may entice them to take those final steps to enter the store. LBA offers marketers great opportunities. Not only are companies able to connect with consumers on a more direct level, but can also supply them with relevant and personalized information. However, with this opportunity comes a responsibility to orchestrate user’s personal
information for profit carefully and intelligently, to make sure it does not raise major concerns and drive customers away from businesses.

### 2.3.1 Spam

Although LBA promises the right information at the right place, it overlooks the relationship, relevance, and preference of the user in regards to the information the advertisers are providing, in favor of short-sighted metrics. As mobile computing continues to grow and become more sophisticated, so does location based targeted advertising. Contents need to be more personalized and relevant to attract customers. If ads are pushed to every customer based solely on their location, it might flood their mobile devices with irrelevant ads and turn them against such applications. It is crucial to understand contextual information in addition to location to make these ad campaigns more effective. According to Kerr, “Location enablement is not scary when it is used to deliver relevant and meaningful offers to an audience that is uniquely prequalified to be interested in the message by where they are at that exact moment” [28]. Fortunately, the solution does not lie too far ahead. Most of the other relevant information useful in tailoring ad campaigns is easily accessible such as time, weather, and user preferences. One of the goals of this thesis is to explore and orchestrate such technique to provide information not just at the right place and time, but also to the right person.

### 2.3.2 Privacy

One of the major advantages in the increasing number of smartphones for advertisers is an opportunity to utilize more targeted marketing based on collected psychographic and demographic information, location, and lifestyle preference. One of the largest challenges of context based smartphone technologies is dealing with privacy issues that worry users continuously [29, 33, 44, 45, 46, 47, 48].

Most LBA applications involve tracking a user’s location and other personal information. Kang [31] explains how an adversary can determine a user’s habits and places of interest which the user visits frequently and other similar personal information solely based on the user’s previous locales. In their experiment, they used an application called The Place Lab [32] which allows a WiFi-enabled client to determine its location by listening to
access points and identifying positions from clustered areas of location samples. With their algorithm, they were able to determine where a user spent most of their time throughout the day with 84% precision. In addition, one can also make inferences about a user’s medical conditions, alternative lifestyles, or political views gleaned from visits to clinics, entertainment venues, or political rallies [33, 34].

This information can be very beneficial for advertisers as they can now provide more personalized advertisements to their customers. Down the road however, it creates concerns among consumers worried about their personal privacy, which can lead to serious consequences if the integrity of the collected data is breached by an unauthorized third party. Exposure of this information can lead to targeted spamming, fraud, stalking, and even physical harm. Laws are being passed restricting advertising agencies from collecting personal data and other private information [30]. According to Gerber [36] “Now, the mobile device feels very private, and location services or push messaging can seem intrusive. It is not just the medium; it is the medium combined with the message and moment—if you screw up any of these elements in relation to the other, it can become not just an effectiveness issue for the marketer but also a privacy issue for the consumer.” Clearly, it is critical for marketers to ensure that they have physical, electronic, and procedural safeguards to protect the customer information they collect to maintain their reputation in the market and their customer’s peace of mind.

2.4 IBM Worklight

IBM Worklight is an open and comprehensive platform for building applications for mobile devices such as smartphones and tablets. The motivation is to provide developers with a single platform to manage the entire application lifecycle from development through back-end integration, authentication, and management after deployment.

2.4.1 Overview

Development of a Smartphone Application

The Eclipse-based IBM Worklight Studio provides a rich environment to develop native, web, and hybrid applications, or apps for short, across multiple mobile platforms including Android, iOS, BlackBerry, and Windows Phone. Worklight’s optimization
framework provides a unique way to integrate a rich user interface into these applications, customized for each platform to match the user’s expectations and styling requirements for all the above mentioned platforms. Despite providing portability, Worklight maximizes the use of a common code base for all environments which saves time, effort, and costs.

Integration of Applications and Other Services
IBM Worklight’s server architecture and adapter paradigm simplifies the integration of applications with back end services, web services, and cloud based services. The Worklight server is a scalable Java based server that can be seamlessly integrated into an organization’s IT infrastructure. In addition, its adapter technology separates concerns cleanly and supports two types of data exchange mechanisms—Device Request and Push Notification.

Runtime Support for Worklight Applications
Worklight Studio provides executable application files for targeted platforms that can be deployed to their respective public app stores and private distribution repositories. Besides that, it provides its own app center to which these applications can be uploaded for distribution. The Worklight server ensures a highly secure and optimized mechanism for communication between apps and services running on the server.

Management of Deployed Applications
IBM Worklight provides support for post-deployment management of apps including monitoring, access control, direct update, and disabling of apps from a single centralized web interface called Worklight Console. Administrators can use it for analytics and reporting as they keep track of usage data.

IBM Worklight Components
The IBM Worklight platform comes with five main components: IBM Worklight Studio, Device Runtime, IBM Worklight Server, IBM Worklight Console, and IBM Worklight App Center. As shown in Figure 5, four of these are the building blocks to orchestrate application development.
2.4.2 Studio

IBM Worklight Studio is an eclipse based integrated development environment (IDE) that allows the developer to perform all the coding and integration tasks to develop a mobile application using four distinct approaches: native, web, hybrid (web), and hybrid (mix). The main features of these IBM Worklight Studio approaches are outlined below.

Figure 5: Overview of IBM Worklight components [12]
Cross-Platform Support
This strategy allows developers to build rich hybrid and native mobile applications on Android, iOS, BlackBerry, and Windows Phone platforms. Using its optimization framework, Worklight delivers a solution which maximizes sharing of common code across multiple platforms without compromising the user experience and functionality inherent in these different platforms.

A developer can customize applications for different mobile platforms while maximizing code reuse using Worklight Studio’s unique file structure as shown in Figure 6 above. In addition, developers can directly access the application programme interface (API) offered by the device as well as integrate third-party libraries, frameworks, and tools such as jQuery templates.

Hybrid Coding
Given the ever-evolving and fragmented ecosystem of mobile devices, application development has become extremely time consuming and expensive. Most organizations today face the challenge of choosing one of the following approaches when building a smartphone application:
(1) Web Applications
These are standard web applications exclusively written in HTML5, JavaScript and CSS to run using the device’s embedded web browser. Because of the “platform-independent” nature of HTML5, it is possible to run typical web applications on any platform that supports HTML5, and most of the mobile devices come with native web browsers that support HTML5. Mobile web applications are extremely popular because of their “write once, run everywhere” nature. The disadvantage inherent in this method is that they cannot access all of the APIs and resources available on the device.

(2) Native Applications
Native applications are platform-specific applications. They are extremely powerful because they have complete access to all available resources and APIs instead of being dependent on the browser’s rendering engine. Moreover, they deliver the highest level of user experience. The downside to native applications lies in the resources and skill sets needed to create them. Table 1: App development across different mobile platforms illustrates technical expertise required to develop applications on various platforms.

<table>
<thead>
<tr>
<th>Table 1: App development across different mobile platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Languages</strong></td>
</tr>
<tr>
<td>Objective-C, C, C++</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
</tr>
<tr>
<td><strong>Packaging Format</strong></td>
</tr>
<tr>
<td><strong>App stores</strong></td>
</tr>
</tbody>
</table>
These cross-platform differences result in much higher costs to develop an app that delivers advanced features and performance.

Both of these approaches have pros and cons. Portability and user experience must be weighed when planning the application. To eliminate this trade-off and achieve the best of both worlds, IBM Worklight offers a different approach; hybrid apps that are a fusion of Web and Native methods. Developers can call native code from HTML based pages to combine HTML and native pages in the same application as well as display HTML and native components together. Based on their structure, these hybrid apps can be categorized as Hybrid Web Applications and Hybrid Mixed Applications.

(3) Hybrid Applications (Web)
The source code of this type of application consists of a web component that executes within a native container provided by Worklight. Thus the application is able to leverage native libraries.

(4) Hybrid Applications (Mixed)
This approach leverages the rich features of web technology along with native APIs by augmenting the web code with native code within the native shell.

Both of the above hybrid approaches give web applications access to the hardware and native APIs such as AR and NFC.

All four approaches are suitable for different scenarios as depicted in Figure 7.
Runtime Skin
The Worklight optimization framework offers runtime skinning which can be applied to the mobile app during run-time. This automatically adjusts the app for different devices with different screen sizes and resolution of the same operating system.

Support for Web Technologies such as HTML5 and JavaScript
The advent of HTML5 has revolutionized the web development domain, opening new doors to creating scalable, ubiquitous software applications that were traditionally hardwired to their respective platforms. Worklight’s support of HTML5 has given developers the freedom to write HTML5 code directly into the app without the limitations of proprietary interpreters and code translators. It also provides many rich capabilities such as geo-location, offline storage, and access to rich media. IBM Worklight enhances existing capabilities with more enterprise level features such as on-device encryption and offline user authentication. On top of that, it also enables integration of third-party JavaScript toolkits and UI frameworks such as jQuery Mobile, Sencha Touch, and dojox.mobile. Developers can also integrate any other free JavaScript libraries or UI frameworks such as jQuery templates.
Native Device SDK Integration
IBM Worklight Studio allows integration of a native shell that enriches the Worklight program with native code capabilities and best-in-class development tools, with the testing and debugging mechanisms that are indigenous to mobile devices. To achieve that, Worklight Studio can be integrated with the SDKs of other mobile platforms.

Standardized Data Retrieval
IBM Worklight Studio uses XSL transformation for hierarchical structured data retrieval from any back-end service in JSON format. This standardization allows developers to invoke back-end services from within Studio and receive responses. Furthermore, it allows back-end services (adapters) to retrieve data from the server and then streamline it to the device. This reduces the number of requests between device and server and hence improves responsiveness of the device.

Unified Push Notification
Besides device request, push notification is another medium of communication between the device and server-side services. Worklight’s unified push notification architecture eliminates interference from a mobile vendor.

Distributed development
Software development processes are comprised of various steps performed by different teams such as development, design, testing, and management. Sometimes these teams operate from geographically different locations. The IBM Worklight platform is designed to fully support such scenarios.
Shell Approach

IBM Worklight follows a shell approach to reduce internal barriers to mobile development and make it ubiquitous across an organization by compartmentalizing skill-sets and responsibilities [43].

![Diagram of IBM Worklight's Shell Approach](image)

**Figure 8: IBM Worklight’s Shell approach [12]**

The customizable native shell provides modularity by breaking down the development into two portions: outer shell and internal application as illustrated in Figure 8. The Worklight application consists of inner modules that provide access to the native libraries and capabilities of the device. The outer shell complies with corporate policies, and the inner app that executes within the outer native shell can be built using nothing but web languages such as HTML5, CSS3, and JavaScript.

Developers can create different shells, each carrying different policies and characteristics to suit different platforms and devices. The common inner application running within
these shells will automatically adjust and comply with outer shells and provide customized UI tailored to satisfy the needs of different devices, promoting code reuse.

IBM Worklight SDK
The software development kit (SDK) provided by IBM Worklight provides the following set of tools and libraries that support the development process:

- JavaScript client API, providing access to Worklight services, environment-specific features and a set of cross-platform user interface utilities.
- JavaScript server API that provides pre-computation capabilities to the server before sending requests or responses to other services or devices.
- Offline access API
- Server and client side unified push notification API
- Localization API to determine the language used by the device
- Schemas for runtime validation for adapters
- Complete tutorials on development, integration processes, and all APIs.

2.4.3 Server
IBM Worklight Server is a Java-based server that serves as a scalable nexus for applications, external services (such as web services), and enterprise back-end services (such as databases). It uses the adapter paradigm to expose the end user device to cloud-based services or legacy systems. This server features three types of adapters: SQL adapter, HTTP adapter and cast-iron adapter that can be used to retrieve and update data from information sources as well as allow end-users to perform transactions and invoke other services and applications. The main features and advantages of IBM Worklight Server include:

Universality
IBM Worklight Server supports multiple integration technologies and back-end information system. It allows developers to integrate several data streams from various
sources efficiently into one and present to the app. As a result, the overall data traffic is considerably reduced.

Scalability
Adapters provide a seamless and efficient way to integrate as many back-end information systems as a developer desires. In addition, it can also support hundreds and thousands of users through physical clustering.

Security
It provides flexible security architecture by integrating back-end authentication infrastructure based on Java Authentication and Authorization Service (JAAS) concepts.

Transparency
The data retrieved from back-end applications is exposed in a uniform manner regardless of the type of adapters.

Data manipulation
It enables developers to add custom server-side logic to manipulate data before delivering to its users.

Server-sided safeguards
Adapter technology adopted by this framework enforces use of prepared-statement for any transaction from and to back-end systems. It helps prevent SQL injection and help protect against cross-site request forgery (XSRF).

2.4.4 Device Runtime
IBM Worklight Device Runtime is a client-side runtime that allows for embedding server functionality within the targeted environment. This runtime consists of client APIs integrated into the native application that complements Worklight server by exposing a predefined and consistent interface between the native application and server back-end irrespective of the platform. Besides that, this runtime provides APIs for on-device encryption, offline authentication, reports and analytics, and remote disabling of apps.
2.4.5 Console
IBM Worklight Console is a web-based user interface allowing developers to monitor and manage the Worklight Server and its deployed apps, adapters, and push notifications. It is used to start/end and enable/disable these server side and client side services. User statistics collected by this component can be used company-wide for reporting and analytics. Furthermore, it provides an efficient way to test user interfaces across different devices through a single platform. Its built-in web-based simulator lets the developer showcase applications across all the different platforms with varied configurations and orientations.

2.4.6 AppCenter
IBM Worklight AppCenter is a private cross-platform app store. It allows collaborative development and orchestration of deployed apps. In other words, it acts as a repository of mobile applications where the developers can upload apps for platforms that are supported through a web interface.

2.4.7 Limitations
IBM Worklight environment offers many features to develop mobile application. The five different components of this package provide many advantages that can be leveraged over traditional mobile application development methodology. However, while developing our application prototype “I-Shop” we identified the following infrastructure limitations.

- Launched in August of 2012, IBM Worklight does not yet have a large user community. As a result it can be difficult to seek solutions to problems a developer might face while working with it.
- The main HTML page created by IBM Worklight Studio involves some hardwired styling which cannot be changed. For instance, the CSS class name of the main HTML file is called “content” which is used in its underlying code to identify the page and invoke necessary procedures.
- It does not provide API support to invoke HTML pages or server side services from native pages.
• It can be integrated with a native SDK to provide most capabilities, but not all of them. For instance, Android’s device emulator allows developers to simulate latitude and longitude values to test their app, a function not supported by Worklight.

• IBM Worklight Studio can only be integrated with IBM product lines including Eclipse and Rational Application Developer (RAD). It does not provide plug-ins for any other IDE such as Microsoft Visual Studio or Netbeans.

• It features a rich editor that enables developers to create HTML pages just by dragging and dropping DOM elements; however, it is not as rich as other tools such as Microsoft Web Expression.

• Most of the SDK for mobile application environments allows data emulation for testing and debugging purposes. For example, Android’s GPS emulator allows developers to emulate the latitude and longitude value for the target device manually for testing. However, the IBM Worklight run-time environment does not provide such functionality which is a significant limitation for developing location based applications.

2.5 Summary
This chapter introduced the reader to context and context-aware applications, particularly location based services and their implications across various domains. As this thesis focuses on location based advertising techniques, it concentrates on one primary application domain of location based services—location based advertising on mobile devices, including the main concerns regarding this technique. Lastly, it discussed the state-of-the-art IBM Worklight infrastructure, a revolutionary product to build smarter mobile applications. We also discussed the advantages and limitations of the IBM Worklight infrastructure identified while building I-shop.
Chapter 3
I-Shop: A Context-Aware Shopping Advisor

Our key contribution is the generic application model. This model is implemented within our prototype application I-Shop—a hybrid mobile application created with IBM Worklight Studio. I-Shop is a context-aware shopping advisor that runs on a user’s mobile device. Unlike traditional location based advertising (LBA) applications, I-Shop is a lightweight, easy to use, dynamic, context-aware, and highly personalized smartphone application that recommends deals and offers regarding nearby products and services. This chapter discusses the key design features of I-Shop.

3.1 Key Features of I-Shop
One of the goals of our application model is to alleviate existing limitations in LBA methodologies and technologies. To gain experience and evaluate our approach, we developed I-Shop. Customization and in particular personalization is one of the main features of our approach. We realize this feature by exploiting user context while still providing a safe and private environment for their data.

3.1.1 Lightweight Client Application
Complying with Worklight’s shell approach, at I-Shop’s heart is the common web component which implements functionalities of the web technology standards HTML5, JavaScript, and CSS. The user interface of this application uses common web controls, such as text view, dropdown lists, and buttons. With these resources, we can provide the user with a powerful cross-platform environment. When running on the device, the application user interface (UI) collects user shopping preferences, such as shoes, cosmetics, or electronics. These preferences are then dispatched to back end services using JavaScript Object Notation (JSON). All data is processed on the back end and the client application receives a minimal amount of information. For data exchange between the different components of I-Shop, we chose JSON because it provides a simple, lightweight data format for the exchange of information [8].
3.1.2 Proactive and Reactive Approaches to User Recommendations

I-Shop uses two approaches to recommend relevant deals to the user: proactive and reactive.

In reactive mode, I-Shop reacts to the user’s input and location. Whenever the user is close enough to a participating business, I-Shop’s back-end services retrieve a list of commodities that are currently attached to special deals. The user is then asked if they are interested in purchasing any of these commodities. If the user responds in the affirmative, they are then provided with a list of advertisements pertinent to the selected commodity available in nearby stores.

The proactive approach provides users with a semi-automatic way to predefine their shopping interests within a wishlist. To populate a user’s wishlist we leverage existing contextual information from Facebook which has an interesting and useful feature called Like. A user can show interest in content, such as movies, books, music, or any other kind of product or service using this feature. The Facebook API allows developers to retrieve a user’s context, which can also include a list of content that they Liked. Our prototype application thereby retrieves a user’s Liked contents from their Facebook profile to build an editable wishlist as shown in Figure 9. Once this wishlist is created, a feedback loop is implemented between the client app and back-end services. These services continuously check for and generate notifications regarding existing deals regarding items on their wishlist when it is appropriate.

![Figure 9: Dialog box for Wishlist creation in I-Shop](image)
3.1.3 Cross-Platform Support

I-Shop has been developed and modeled using IBM Worklight optimization infrastructure. This allows development of mobile applications that use core web technologies embedded into the native shell of most major smartphone operating systems, including Android, iOS, BlackBerry, and Windows Phone. Thus, a developer only needs to write the code once to run everywhere. The prototype application developed for this thesis operates on the Android platform, but can potentially be deployed in other environments by following two following simple steps.

1. Download and install the software development kit for the desired platform (e.g., iOS, BlackBerry, or Windows Phone) to be supported.
2. As shown in the screenshot in Figure 10, one can easily embed the common web module of I-Shop into any environment using IBM Worklight infrastructure. The software will then output an executable file capable of running in the respective environment, including not only four major smartphone operating systems, but also desktop or web environment.
Figure 10: Adding different environment to a single application

3.1.4 Personalization
I-Shopper utilizes the following data to create a more personalized user experience.

- The user’s personal context is retrieved from their social networking profiles.
- Her current location.
- Her preferences to filter deals offered. For instance, if her interest is shoes, all the deals except for shoes will be filtered out before presenting the results to her.

For instance, the shopping wishlist of a user is appended with products and services they Liked on Facebook. The user interface of I-Shopper is highly customizable. Based on the
user’s gender, the logo of I-Shop is modified. Figure 11 shows the two different logos used.

![Custom I-Shop logos for female and male users](image)

Figure 11: Custom I-Shop logos for female and male users

Moreover, users are greeted with their picture at the top of the I-Shop UI to provide an extremely personalized experience to its users, hence the ‘I’ in I-Shop.

### 3.1.5 Content Generation
I-Shop uses two sources to gather information on deals and offers: Groupon and its own extended database. Groupon provides free API support to developers to build their own applications. I-Shop harnesses the power of Groupon's data infrastructure and collects available deals in the user’s current city. Our extended deals database also contains a list of deals across various locations. These deals are intended to be recorded into the database by the merchants themselves. Present information in the database has been simulated to showcase the proposed model.

### 3.1.6 Statelessness
Unlike many other advertising applications for smartphones, I-Shop never tracks the user’s location to orchestrate system behavior. With the help of feedback loops, we use a reactive approach where the application is triggered only when the user’s current location
is within a defined proximity of a business that is providing relevant deals. People can argue that our approach exploits the user’s personal information retrieved from Facebook for personalization. However, we do not record the data permanently. In fact, our application model provides a mechanism where the user’s personal information is held only on their handheld devices for the current session. Our application directly communicates with Facebook’s API without any involvement of its back-end services. That means only the user’s device knows their personal information and nothing else. Moreover, I-Shop can only obtain this information if the user agrees to share it by logging into their Facebook account.

3.1.7 Self-Updating
Unlike most smartphone applications where the user needs to manually download and install the updated version, I-Shop leverages IBM Worklight’s direct update feature to eliminate the aforementioned task. Instead, this smart application checks for updates and performs the necessary operations by itself upon start-up. IBM Worklight’s Direct Update feature lets developers update the web content of their deployed web and hybrid applications directly from the IBM Mobile Application Platform Server upon application start-up. Figure 12 demonstrates the model which makes updating apps autonomous by orchestrating feedback loops in the following four steps.

![Figure 12: IBM Worklight direct update architecture][12]
1. The app packaged with web resources is downloaded and installed on a smartphone.
2. Web resources are then cached to local memory for offline access and better performance.
3. Upon starting the application, a feedback loop is triggered to check for available updates.
4. If updates are available, the native shell downloads the updated app and deploys it at the same time.

I-Shop also enjoys direct update features that gives the user freedom from having to manually update their app after every modification.

3.1.8 Internationalization
Smartphones are ubiquitous and so are smartphone applications. To be useful to human beings across the globe, these applications should be able to self-adapt to changing requirements and personal contexts such as different dialects. I-Shop realizes this unique feature by leveraging IBM Worklight’s efficient, yet simple, mechanism to translate text into different languages. When the user logs into Facebook via I-Shop, its context mining module determines their language and provides an option to translate into their language. This feature not only makes it more usable but provides a more personalized experience.

3.1.9 Hybrid Coding
To eliminate trade-offs and produce the best of both worlds between native and web apps, IBM Worklight provides a mechanism that allows a developer to build a hybrid application using either core web technologies such as HTML5 and JavaScript, or a combination of both native and web technologies have it embedded in different native shells that can then seamlessly execute on their respective platforms.
Our prototype application is such a hybrid app, built using IBM Worklight’s infrastructure. At the core of I-Shop lies a web component consisting of HTML, JavaScript, and CSS. It can be reused and infused into any of the Android, iOS, BlackBerry, and Windows Phone platforms using IBM Worklight Studio. It can also be altered to run on a variety of devices including tablets and desktops. The optimization framework allows developers to customize the outer shell and layout of these hybrid apps. As depicted in Figure 13: I-Shop’s file system structure, an IBM Worklight project contains a *common section* with all the common business logic, styling, and markups. Developers can then infuse these common elements into any environment. To achieve maximum adaptation to a specific environment, the core web component of that particular environment can be optimized for it. For instance, an iOS environment requires an iOS-specific look and feel that can be achieved by extending styling scripts and JavaScript files.
In addition to web capabilities, IBM Worklight provides a set of JavaScript APIs to invoke native controls regardless of environment and automatically renders them in a style customized to each platform. Thus, I-Shop features both web and native capabilities. Figure 14 shows how these components interact with each other through the API calls provided by this infrastructure.

![Hybrid App – Interaction with Mobile Device](image)

**Figure 14: Hybrid app: Interaction between web and native components [58]**

### 3.1.9 Scalability

Connecting the app to back-end systems presents multiple hurdles with respect to security, authentication, and scalability. As organizations mobilize more lines of business, expansion of existing services and control of multiple applications has become a daunting task. Our proposed model follows a simple client-server architecture based on the IBM Worklight architecture designed to overcome these challenges. The client app runs on a user’s device and back-end services run on the Worklight server. This Java-
based server provides a scalable gateway between applications and external web services as well as back-end services by means of adapters. Conceptually, an adapter is a set of JavaScript functions that can be remotely invoked by an application. Figure 15 depicts a simplified view of the adapter framework adopted by IBM Worklight.

![Diagram of adapter paradigm adopted by IBM Worklight platform](image)

**Figure 15: Adapter paradigm adopted by the IBM Worklight platform [12]**

This framework provides scalable and reusable end points for connectivity and multiple-source data extraction and manipulation. An app running on a smartphone can invoke procedures in an adapter. The adapter then retrieves the requested information from data sources, manipulates it, and then returns it to the client app.

I-Shop uses two back-end adapters, the HTTP and SQL adapters. These two end points allow seamless integration of multiple data sources. These services currently extract information from two data sources, Groupon and our extended deals database, and the framework provides us with a scalable solution to incorporate more data sources in the future. To achieve this, we only need to write a small procedure in one of these adapters. Figure 16 presents sample code for an I-Shop’s HTTP adapter collecting deals from various sources, concatenating them, and dispatching the information to the client application.
3.1.10 Dynamic Control

I-Shop exploits IBM Worklight’s unique feature of online support to orchestrate deployed apps. Figure 17 is a screenshot of its administrative console, where developers can monitor the system’s components. It also allows them to disable any deployed application remotely for any particular platform or device type. Developers can also send custom messages to end users through push notifications deployed in their administrative control. In other words, the end user can be notified with custom messages regarding any changes made.

```javascript
function getDealsInArea(lat, lng, selectedItem, currentArea){

    // Get deals from extended deals database
    var dbData = WL.Server.invokeProcedure({
        adapter : 'sqlAdapter',
        procedure : 'getDealsInArea',
        parameters : [currentArea, selectedItem]
    });

    // Get deals from Groupon API
    var grouponData = getGrouponDeals (lat, lng, selectedItem);
    var finalResult = grouponData.grouponDeals.concat (dbData.resultSet);

    return {"result": finalResult}
}
```

Figure 16: Groupon HTTP adapter for I-Shop
3.2 Scenario Based Interaction Model for I-Shop

To describe the interactive model of I-Shop best, we adopt scenario based design as developed by Carrol [60]. Figure 21 illustrates a typical use case scenario for our prototype involving the following steps.

- While the user is online on Facebook, she launches I-Shop on her mobile device. If the user is not logged into Facebook, she is prompted with a dialog to login as shown in Figure 18.
After the user has logged in via Facebook, I-Shop invokes a client side service to retrieve the user’s personal context from her/his Facebook account. This information includes her gender, locale, and the list of products and services that she has shown interest in. We use this information to personalize the UI of our app. As depicted in Figure 19, since the end user is female, we display a female-oriented logo. In addition, our UI welcomes our user with her name automatically retrieved from the Facebook API.

After UI customization, I-Shop offers her to translate I-Shop into her native language (Figure 20), which is also retrieved from Facebook. If she confirms, the custom messages are then translated into her native language.
Figure 21: I-Shop's scenario based interaction model
There are two approaches I-Shop reacts to user’s proximity to the shopping area. First is proactive approach in which user creates their own shopping wishlist. As shown in Figure 22, a modifiable list of products and services retrieved from the Facebook API is presented to the user to enable her to create her own wishlist where she can have hundred percent control to delete existing items or add more. Thereby, I-Shop constantly checks for deals on these items in user’s proximity.

The second approach is reactive approach where user is presented with a list of generic items to shop for depending solely on their location. As shown in Figure 23, user can thereby select any domain to view the deals in current area.

On item selection, the selected item is sent to the back-end services to search for deals close to their current area. If any deals currently exist regarding the selected item, the
user is presented with a list of such deals across different stores in that area as demonstrated in Figure 24.

- The user can further click on the deal to view additional details on a vendor’s website. Figure 25 is a screenshot of such detailed view.

![Figure 24: I-Shop user interface displaying deals](image1)

![Figure 25: Sample deal offered on a store's website](image2)

### 3.3 Summary

In this chapter, we discussed key architectural and functional features of our application model. The scenario based design reflects an abstract representation of the high-level functionality of our designed I-Shop prototype, whose main objective is to provide the user with a personalized, context-aware environment where she does not need to worry about privacy issues. Besides this, we also aim is to using IBM Worklight infrastructure to lower the overhead for building smartphone applications by providing cross-platform support.
Chapter 4
I-Shop Data Sources and Application Model

Our key contribution is the application model for implementing context-aware location based advertising techniques with cross platform support, statelessness, and internationalization. This chapter overviews key data sources for this application, explains how these sources are used in I-Shop, and describes our I-Shop application model in detail.

4.1 Data Sources of I-Shop

4.1.1 Groupon

As defined by Google Finance [49], Groupon, Inc. is a local e-commerce marketplace that connects merchants to consumers by offering goods and services at a discount. Each day, the company emails its subscribers a list of discounted offers for goods and services targeted by location and personal preference. Consumers also access deals directly through Groupon’s website and mobile applications. Today, Groupon has approximately 50 million subscribers in over 560 cities across the globe and is one of the most popular distributors of e-coupons [50].

Groupon provides API support to developers, allowing integration of deals into third-party applications. This API provides two services: Divisions and Deals.

- The Divisions API is responsible for returning information on Divisions supported by Groupon, where a Division is defined as a logical grouping (i.e., geographic or non-geographic) of deals that may appear on the Groupon site. Currently, divisions represent discrete metropolitan areas.
- The Deals API returns an ordered list of deals currently offered in a given division. This list is sorted based on position data within the request (i.e., latitude and longitude coordinates of the user).

Both of these APIs support JSON as well as XML.
4.1.2 I-Shop and the Groupon Deals API

I-Shop harnesses Groupon’s data infrastructure by using the Deals service to retrieve offers available in close geographic proximity to the user. To obtain this information, an HTTP get request is issued to the web service at http://api.groupon.com/v2/deals with the following URL parameters:

- **Client_id**: The API key that identifies the client (developer)
- **Lat**: Latitude of the location
- **Lng**: Longitude of the location
- **Show**: A list of comma separated values to be returned in the response object
- **Merchant**: Information about the business providing the deal

![Figure 26: Screenshot of a sample deal on www.groupon.com illustrating the values used by I-Shop](image-url)
As illustrated in Figure 26, the following five pieces of information regarding deals can be requested by I-Shop:

- **title**: The title of the deal
- **smallImageUrl**: URL of the icon-size image for the business being queried
- **dealUrl**: URL where the deal can be viewed in detail
- **announcementTitle**: Title of the deal

I-Shop retrieves these details in JSON format. The response depicted in Figure 27 is a sample response as sent by the `getGrouponDeals` method implemented in the server side HTTP adapter of I-Shop that retrieves deals from the Groupon API. This information was retrieved in response to the following query parameters to the `getGrouponDeals` method:

- **Latitude**: 48.336 (Latitude value of the user’s location)
- **Longitude**: -123.3648 (Longitude value of the user’s location)
- **Item**: clothes (Name of the product that the user wants to see deals on)

```json
{
   "grouponDeals": [
      {
         "deals": "$29 for a 24-Pack of iChill Relaxation Shots ($75 List Price). Free Shipping.",
         "details": "$29 for a 24-Pack of iChill Relaxation Shots",
         "link": "http://www.groupon.com/deals/ichill-relaxation-shots",
         "storeName": "iChill Relaxation Shots"
      }
   ],
   "isSuccessful": true
}
```

**Figure 27: getGrouponDeals response in JSON format**

Thereafter, these deals are filtered to display only items the user is interested in. These deals are then augmented with deals retrieved from other sources and presented to the user on her smartphone.
4.1.3 Facebook

Social Networking Sites (SNS), such as Facebook, Google+ and Twitter, provide an innovative platform for social interaction and communication. Launched in 2004 by Mark Zuckerberg, Facebook is now the world’s largest social network with more than one billion users [51]—more than half of them accessing it from their mobile devices [52]. Like most other social networking websites, a user must sign in and create a personal profile. From the popularity gained through the years and its vast user base, it is clear that people are willing to give away selected personal information to socialize with friends. Figure 28 exhibits the types of personal information stored by Facebook.

![Figure 28: Personal information stored by Facebook](image-url)
In exchange, a user is given privileges to add friends, exchange messages, as well as broadcast and share information with text, images, and videos.

As a result, this huge web of social networking has accumulated a rich set of contextual information that can be leveraged not only to provide the user with relevant information, but also to help engage their commercial interests. This information can be in the form of advertisements that not only benefit the user as she is receiving information aligned to her interests and characteristics, but also vendors as they can now focus on those people who most likely can be influenced by the ads they are shown.

The Facebook platform provides a variety of APIs and tools that enable third-party developers to build applications for Facebook members. These APIs allow applications to use social connections and profile information to make applications more engaging and publish activities to the news feed and profile pages of Facebook, subject to an individual user’s privacy settings [55]. With the API, a user can add social context to her applications by utilizing profile, friend, page, group, photo, and event data. The API uses the Restful protocol with localized responses in XML or JSON format [56].

The Facebook platform offers a variety of APIs to serve different purposes. This thesis aims to leverage personal context from social networking profiles to personalize deals that are being offered and to improve the overall user experience. The Facebook platform offers the following four ways to access a user’s personal data.

**Facebook Login API**

It simplifies the process of connecting the user with the application by providing a Facebook login dialog [56]. Figure 29 depicts a screenshot of such a dialog box which creates a trusted link between the app, its user, and her information. Once the user logs into the app using her Facebook credentials, she is prompted to decide how the app will make use of her information. She will see a dialog box explaining what she is sharing, giving her a choice to remove individual permissions, or letting her skip the step completely. She can also revoke permissions in her account settings [56]. Once she has given permission to the app, it can use several methods implemented in JavaScript made
available by Facebook APIs to access her information and build functionalities based on it.

![Facebook login dialog box](image)

**Figure 29: Facebook login dialog box**

**Facebook Graph API**

At the heart of Facebook is the social graph representing a simple, consistent view of the Facebook social network, uniformly representing objects in the graph (i.e., *people, photos, events, or pages*) and the connections between them (i.e., *friend relationships, shared content, or photo tags*). Facebook Graph is a low-level HTTP-based API that developers can use to query data, post new stories, create check-ins, or any of the other tasks that an app might need to perform. Facebook's Open Graph allows developers to create new objects and actions in a user's social graph. The way developers create new instances of those actions and objects is via the Graph API. However, to get additional info beyond that, the developer must first get the user's permission using the Facebook Login API.
Facebook Query Language (FQL)
Facebook Query Language or FQL is an SQL style interface provided by Facebook to query the data exposed by the Graph API. FQL provides some advanced features such as batch queries that allow running multiple queries at once. The FQL query patterns are similar to SQL patterns. Figure 30 depicts an FQL query that takes a user id (uid), the logged user, and returns her name.

```
SELECT name FROM user WHERE uid = me()
```

Figure 30: Sample Facebook query

Facebook Legacy REST API
This is the old REST API supported by Facebook that enables developers to interact with the Facebook website programmatically via HTTP requests. Similar to the Facebook Graph, this API provides support for all kinds of conventional interactions such as publishing events, ads, and posts on the user’s wall or just general information access. However, this API will soon be deprecated as Facebook will be using the new Open Graph API [56].

I-Shop and the Facebook Login API
I-Shop uses the Facebook Login API to connect with its users. For authentication and authorization it uses the OAuth 2.0 standard [56]. OAuth 2.0 is the new version of the OAuth protocol which focuses on simplifying the task of establishing connectivity between the client and the application while providing specific authorization flows for web and desktop applications, mobile phones, and living room devices [59].

This method requires the user to login with the Facebook dialog using her credentials and give permission to I-Shop to obtain her information. The Facebook login includes several security features to protect its users, but it also allows third-party apps to request permissions from users in a consistent flow across all devices. Whenever a user logs into I-Shop using this API, she is prompted with a login dialog. Upon authorization she is presented with a dialog containing information about which permissions the app needs.
and what the app does with these permissions. If she grants permission, I-Shop extracts the contextual information to personalize itself.

Facebook login follows an access token identification process where a token is a random string that provides temporary, secure access to Facebook APIs. This token is used to identify a user, app, or page session and provides information about granted permissions. Since I-Shop aims to retrieve the user’s personal information, it uses a *User Access* type token. This token type is the standard type for API calls and is generated in the login flow when the user grants permissions to an app. I-Shop uses this token to perform API calls on behalf of the user, including reading personal information. The access token identification process is intended to secure usage of applications to only those who are trusted. The token should be kept secret as it enables applications to make API calls on behalf of the user.

![Facebook login architecture](image)

**Figure 31: Facebook login architecture [56]**
The Facebook API provides a JavaScript API for JavaScript-capable apps, including a quick way to integrate login flow into an app for any device with a JavaScript-capable browser. This SDK handles authentication and authorization. Figure 31 presents a two-step high-level architecture demonstrating the login process of a user into I-Shop using the Facebook API.

1. The user clicks on the *Login with Facebook* button triggering the login process and showing her the *Login Dialog* where she can enter her credentials.
2. A user access token is generated and then used by I-Shop to make API calls and retrieve contextual information. Access token persistence and signing of API calls are handled automatically by the client-side JavaScript SDK.

![Interaction architecture between Facebook and I-Shop](image)

*Figure 32: Interaction architecture between Facebook and I-Shop*
As shown in Figure 32, our application model utilizes Facebook context in two different ways:

1. Personalization of the user interface and internationalization. It retrieves the following information from the Facebook API:
   - id: The user’s Facebook id used to access her profile picture
   - firstName: The user’s first name
   - name: The user’s last name
   - gender: The user’s gender: male or female
   - locale: The user’s locale (native language)
   - languages: Array of objects containing language id and name

2. To build the user’s shopping wishlist, we extract the list of the products and services the user has liked on Facebook. With the user’s consent, this list is then used to create a shopping wishlist.

**Extended Deals Database**

Besides Groupon, another source of deals is I-Shop’s custom database built using MySQL. It consists of various offers across various stores specializing in a variety of commodities.

As depicted in the ER diagram in Figure 33, our extended deals database consist of three main tables: ShopLocations, Deals, and DealLocations.

**ShopLocations**

This table records the location information about the stores including street address as well as latitude and longitude values. This table is queried against the user’s current position to determine if she in close proximity to any relevant shopping location.
Figure 33: ER diagram of our extended deals database

Deals
All the information regarding the various deals across the stores is recorded here including short and long description of the deal, name of the item on the deal, name of the store, as well as URL and image of the deal. Besides that, it also contains related keywords that are semantically similar to the item.

DealLocations
This table creates a relationship among the other two tables – Deals and ShopLocations. Each record in this table represents a unique deal in a particular location. This table is queried against the user’s location and selected item to retrieve deals from the Deals table. In addition, it also contains the time period within which the deal is offered.

Our extended deals database is directly exposed to the SQL adapter running at the server. Actually the SQL adapter forms another transport layer between the HTTP adapter and the database. The reason behind sending the deals retrieved from the database to the
HTTP adapter instead of the client is to leverage the HTTP adapter’s existing logic to manipulate data before presenting it to the end user.

### 4.2 I-Shop: Application Model

Figure 34 illustrates the high level overview of our prototype application I-Shop. Our approach is to customize application behavior based on personal context to improve user experience. This application is built around a simple client-server architecture where the client is a hybrid application running on the user’s smartphone and the server is a transport layer comprising back-end services that react to the client’s requests.

![Diagram of I-Shop application model]

Figure 34: Overview of I-Shop application model
The client application running on the user’s device allows incorporation of personal context fetched directly from Facebook to customize its behavior. The customization logic resides on the client application to eliminate any privacy concerns as this information is then limited to the device only. The user’s location and shopping preferences are the only two pieces of information sent to the back end to retrieve relevant deals. In addition, the UI composition logic is also implemented within the client framework.

At the back-end, we incorporate adapter technology to fetch shopping suggestions from the Groupon API and our extended deals database. As depicted in Figure 34, we incorporate a chain of HTTP and SQL adapters which serve as a transport layer between client and data sources. Once the information is retrieved and filtered by the adapters, the HTTP adapter structures and concatenates the list, then presents them to the client in JSON format.

**I-Shop: Client**

The IBM Worklight infrastructure plays an important role in our approach. It includes a variety of technologies and techniques to support mobile application development, and offers features that our client app uses to build a scalable, powerful, rich, and portable application. The key design aspects of our client are as follows:

**Web-based development**

IBM Worklight apps are developed using standard web technologies including:

- A single main HTML5 file.
- Multiple JavaScript files containing the business logic.
- CSS files and images.

This web component of our client application is then embedded or augmented into the native shell of any environment, which allows us to leverage native controls and capabilities along with web capabilities. It also allows us to seamlessly switch between native pages and web pages that let developers have the best of both worlds. IBM Worklight provides a set of rich JavaScript APIs to implement this. For instance, Figure
35 demonstrates the use of the `WL.NativePage.show()` method provided by the IBM Worklight API to show a native page. Another example is the `WL.Logger.debug()` method that allows developers to log messages and information required for debugging.

```javascript
function openNativePage() {
    var param = {'input': $('#input').val()};
    WL.NativePage.show('com.ishop.HelloNative', responseFromNative, param);
}

function responseFromNative (resp) {
    WL.Logger.debug('Response from native: ' + resp.data);
}
```

**Figure 35: Example of IBM Worklight JavaScript APIs**

**User Interface Controls**

One of the main advantages of being able to use web technologies in mobile computing is common user interface controls. Some of these controls such as modal pop-ups, text boxes, and loading screens are common to all the smartphone platforms. IBM Worklight provides a set of JavaScript APIs to invoke these controls regardless of the hosting environment. As shown in Figure 37, these APIs automatically render them in their native style. For instance, the code fragment depicted in Figure 36 demonstrates a common control dialog window with buttons in three different environments which are implemented using a common API called `WL.SimpleDialog`. Such simple dialog can be implemented with one line of code regardless of the environment.

```javascript
WL.SimpleDialog.show(title, "Would you like to buy something today?", buttons, options);
```

**Figure 36: Implementation of a common WL control**
Our client application enjoys a rich set of common controls such as text box, alert, logger, and modal popup.

Optimization: Environment-Specific Code
One of the main features of our approach that makes it stand out is its optimization framework which promotes code reuse and eases maintenance across multiple environments. Our application is developed using the IBM Worklight optimization framework which gives it the potential to be run in multiple mobile, tablet, and web environments by having written core logic and design guidelines in HTML5, JavaScript, and CSS. Runtime environments differ from each other in many aspects, including screen size, resolution, and user interface controls. To achieve maximum adaptation and perfection on each platform and device, web components can be optimized specifically.
For optimization purposes and separating environments, the application structure is divided into multiple environment folders as depicted in Figure 38. Each folder representing a different environment contains resources relevant to that specific environment. When the application is initially created, it contains only one folder called
common which is the foundation for all environments. Later on, when different environments (e.g., iPad, Android, or desktop) are added, these common resources are extended among them and these files are then overridden and customized automatically by the IBM Worklight optimization framework to adapt to the native environment.

![Figure 38: I-Shop's optimization file structure](image)

**Access to Native Mobile APIs**

Although I-Shop’s core business logic and user interface are built using web technologies, it is still able to access native mobile APIs. This is possible due to integrating the Apache Cordova framework into the IBM Worklight platform itself. Apache Cordova is an open source mobile development framework that provides a JavaScript API allowing developers to access native features of a mobile device and even execute native code.

Currently, the Apache Cordova project provides support for five platforms, including iOS, Android, BlackBerry, Windows Phone, and Bada. However, this framework has
been integrated into IBM Worklight for only iOS and Android. It is automatically augmented to IBM Worklight applications and can be invoked using JavaScript APIs. Some of the most popular Apache Cordova libraries include Geolocation, Device, Accelerometer, and Camera.

Geolocation is one of the most important parts of our application model. The I-Shop client application uses Apache Cordova’s geolocation library to retrieve location information for the device. The main advantage of using this library is accuracy because it not only uses Global Positioning System (GPS) information, but also IP address, RFID, Wi-Fi and Bluetooth MACs, and GSM/CDMA cell ID information.

This particular API provides three main methods: getCurrentPosition, watchPosition, and clearWatch. Our application prototype uses two of these methods: watchPosition and clearWatch.

**watchPosition**
This is an asynchronous method used to watch for any changes in the device’s position. This method is used by I-Shop to determine if the user is in close proximity of any shopping area. If he is, it triggers the search for deals and notifies her of incoming results. The code fragment depicted in Figure 39 involves use of this method and was written for our project.
clearWatch
This method stops I-Shop to watch for changes to the device's location by clearing the `geolocation.watchPosition` referenced by `watchID`. This method is used when an I-Shop user has reached a particular shopping destination. Thus, I-Shop does not need to check her position and unnecessarily utilize resources. The I-Shop code fragment depicted in Figure 40 where this function is used.

```
function getLocation() {
    var watchID;
    if (navigator.geolocation) {
        watchID = navigator.geolocation.watchPosition(
            // Returns the device's current position as a Position object.
            // Success
            function onSuccess(position) {
                // "position" is a Position object.
                // Search if there are any interesting deals in the current location
                searchDeals(position.coords.latitude, position.coords.longitude, wishlistItems);
            },
            // (Optional) Failure
            function onError(positionError) {
                // "positionError" is a PositionError object.
                alert("Error");
            },
            // Options (optional argument)
            {
                frequency : frequency,
                enableHighAccuracy : true,
                timeout : timeout,
                maximumAge : maximumAge
            });
    }
}
```

Figure 40: I-Shop uses the clearWatch method of IBM Worklight's Geolocation API
Modular Dynamic User Interface Composition

Our prototype application consists of a single primary HTML file containing multiple div HTML elements hosting different pages. As shown in the sample code depicted in Figure 41 in our approach we create separate div elements with required styling and render them at runtime. We reuse basic styling (e.g., background logo) to maintain consistency among all pages.

```html
<div id="page1"></div>
<div id="page2"></div>
<div id="page3"></div>
<div id="page4"></div>
<div id="page5"></div>
```

Figure 41: Modular user interface composition

It can be a daunting task to maintain a single large HTML file with multiple pages. To address this issue, we use a modular approach in our prototype and create small separate markup files representing multiple pages which can then be rendered inside these div HTML elements at runtime.

**jQuery Template for Dynamic Display of Data at Run Time**

I-Shop displays multiple deals for the user. These deals are dynamically obtained from the back end in JSON format. We use jQuery templates to render them. This API enables developers to display and manipulate data in the browser [3]. We used this technology to create and render lists dynamically at runtime, including a list of items to be added in a wishlist and a list of deals to be displayed on the user’s device. This technique requires users to create a simple template for one item in a list. For example, Figure 42 depicts the HTML fragment dealListComponent.html, which represents one item in a list of deals that has five elements: *image, title, store name, deal*, and *website URL*. 
Now this single list item can be rendered dynamically using the jQuery code depicted in Figure 43.

```
jQuery.get("dealComponent.html", function(response) {
    WL.Logger.debug(deals);
    window.jQuery("#dealsList").empty();
    jQuery.template("dealTemplate", response);
    jQuery.tmpl("dealTemplate", deals).appendTo("#dealsList");
});
```

Figure 43: jQuery Template code to bind JSON response with template user interface (dealComponent.html)

In this code, the `jQuery.template()` function takes the contents of the `dealComponent` file and compiles it as the named template `dealTemplate`. The `jQuery.tmpl()` function then takes the first element in the matched set and renders its contents as a template using the specified data `deals`. Later on, it finds the DOM element in the main html file with `ID = dealsList` and renders the compiled list as shown in Figure 44.
Figure 44: User interface composition using jQuery templates

Integration with Facebook
From the four approaches mentioned above to collect user data I-Shop uses the Facebook Login API. This API provides a JavaScript SDK for JavaScript-capable apps that include quick integration of login flow into an app for any device with a JavaScript-capable browser. This SDK is responsible for authentication and authorization. It provides developers with a few basic methods that can easily be integrated into any HTML page as follows:

- `FB.login()` — asks the user to login or requests additional permissions.
- `FB.logout()` — log the user out (only if the user has authorized the application).
- `FB.getLoginStatus()` — asynchronous method to retrieve the current Facebook login status of the user.
- `FB.getAuthResponse()` — synchronous access to the current authorization response record.
I-Shop uses Facebook login and logout methods by means of a simple `onClick` event of a button as depicted in Figure 45.

```html
<div id="fb-root"></div>
<script src="js/facebookIntegration.js"></script>
<div>
  <div id="login">
    <p>
      <button class="FbButton" onClick="FB.login();">Login with Facebook</button>
    </p>
  </div>
  <div id="logout">
    <p>
      <button class="FbButton" onClick="FB.logout();">Logout of Facebook</button>
    </p>
  </div>
</div>
<script src="http://connect.facebook.net/de_DE/all.js"></script>

Figure 45: HTML code to orchestrate Facebook login

The SDK also provides several events that can be subscribed to such as the `auth.statusChange` event, which is published when the user logs in or out. We use this event to change the visibility of the login/logout button and also invoke the `contextManager` segment of our application to extract the user’s information.

I-Shop uses the `FB.api` method to extract the list of products and services that the user likes as well as her personal information as depicted in the code fragment in Figure 46.
Figure 46: I-Shop implements methods provided by Facebook API to retrieve personal context

With user consent, this list of liked products and services is then used to create a shopping wishlist. The user is then informed of any current deals being offered nearby on any item in their wishlist.

Internationalization
By leveraging the IBM Worklight framework, our client application can easily translate the application strings and system messages. The IBM Worklight studio creates a messages.js file which is intended to store application strings. Figure 47 demonstrates I-Shop’s messages.js file which contains default strings as well as methods to translate these strings into different languages.
Now these strings can be automatically translated at runtime by invoking any of these methods to override the default messages. In addition to this, we need to specify where these strings are used in the markup which is achieved with two simple steps.

1. As shown in Figure 48 the DOM element displaying these translatable messages should belong to a class `translate`.

2. The ID of each DOM element requiring translation should be same as the name of variable name in `Messages.js`
I-Shop: Server

The I-Shop client application on a user’s mobile phone is in constant interaction with back-end services powered by IBM Worklight Server. These back-end services are responsible for triggering search processes, retrieving deals from data sources, filtering based on the user preferences and location, appropriate formatting, and sending the results to the application. Our approach leverages existing infrastructure provided by IBM Worklight to provide a scalable solution to connect with various back-end systems.

Integration Adapters
As described in Chapter 2, IBM Worklight Server provides adapter technology that connects to a variety of enterprise information systems over widely used integration technologies such as Simple Object Access Protocol (SOAP), Representational State Transfer (REST), Structured Query Language (SQL), and Lightweight Directory Access Protocol (LDAP).

What is an Adapter?
An adapter is a transport layer between a mobile application and external information systems such as web services. They are used to retrieve information from a variety of services and perform an action on the retrieved information. These adapters are capable of combining information from multiple sources into a single response for the client application. In addition, adapters can also be used to cache frequently-requested data.

Each adapter consists of two files:

1. An XML file used to declare procedures and connection properties.
2. A JavaScript file which defines procedures and contains business logic.

A procedure is a JavaScript function that provides functionality to the adapter. The client application can invoke these procedures and obtain results in JSON format.

Adapter types
Currently, the following four types of adapters are provided:

1. SQL adapter (connects to databases)
2. HTTP adapter (supports both REST and SOAP based services)
3. Cast Iron adapter (IBM WebSphere Cast Iron Cloud Integration)
4. JMS adapter (supports the exchange of messages from any messaging provider that supports the Java Messages Service (JMS) API.

How adapters work
As depicted in Figure 49, these adapters follow four basic steps:

1. The client app running on the mobile device makes a call to a procedure defined in any of the adapters.
2. The procedure sends the query to the respective back-end service.
3. When the adapter receives the response back from the service, it either returns it raw or preprocesses it in accordance with predefined logic and then presents it to the application as a JSON object.
4. Finally, the adapter sends the information to the application in JSON format.
Figure 49: Overview of adapter technology [12]
Adapters in I-Shop

Our prototype application uses two adapters to retrieve data from two different sources: the HTTP adapter for Groupon and the SQL adapter for our extended deals database.

- The HTTP adapter for I-Shop serves as a transport layer between:
  - Client app and Groupon API
  - Client app and SQL adapter

- Adapter responsibilities include:
  - Check for any deals being offered within 500 meters of the user’s position.
  - Obtain deals from the Groupon API and SQL adapter based on the user’s position.
  - Filter the retrieved deals based on the user’s current selection and wishlist.
  - Concatenate the list of deals from the two sources into one JSON object with consistency and present to the client.

- The SQL adapter for I-Shop serves as a transport layer between the HTTP adapter and the SQL database. It contains procedures with the following functionalities:
  - Determine if the user is close to a shopping destination.
  - Send a list of items to the HTTP adapter to search deals in our extended deals database corresponding to the user’s current position.

4.3 Summary

This chapter discussed key architectural and functional features of our I-Shop application. In particular, we explained how I-Shop accesses data sources and derives context. We also discussed the design of the client-server architecture model employed by the I-Shop application model including selected components.
Chapter 5
I-Shop Implementation and Evaluation

This chapter begins with an introduction to the implementation in our application model. The focus then shifts to the evaluation of I-Shop, in which our I-Shop prototype is evaluated in an experiment where we assess its usability in terms of efficiency, effectiveness, and user experience.

5.1 The I-Shop Implementation Model
One of the unique features of our approach is the cross-platform support made possible by the IBM Worklight optimization framework. This framework currently supports the following major mobile platforms: Android, iOS, BlackBerry, and Windows Phone. We chose to implement and test this model only on Android for the following pragmatic reasons.

First and foremost, the cost of the development kit and distribution channel for Android is free for developers. Secondly, in accordance with a survey conducted by IDC (see Figure 50), Android has become the dominant platform in the mobile marketplace, so development for it will reach the largest number of users. IDC looked at the global operating system share among these smartphones in the first quarter of 2012 and came to the conclusion that 59% of smartphone owners possess Android-based devices [65].

Figure 50: IDC Worldwide mobile phone tracker as of May 24, 2012 [1]
Lastly and maybe most importantly, I own an Android based smartphone, which allowed and encouraged me to explore the platform with 24/7 access.

As discussed earlier, our prototype application is built using core web technologies. The IBM Worklight infrastructure plays an important role in our approach. It provides cross-platform support through a rich set of custom JavaScript APIs. It allows developers to focus more on their ideas rather than worrying about the time and resources they need to expend to implement that idea on every platform. Besides this, jQuery templates are a large part of our dynamic user interface composition which enables automatic display of JSON responses on the client app from the back end.

Our approach is built around a simple client-server architecture where most of the data manipulation occurs at the server level. The client running on a mobile phone mainly consists of the Context Manager and Facebook Integrator. The Context Manager is responsible for updating the user’s wishlist and personalization based on context. The Facebook Integrator handles context retrieval from Facebook. We chose to expose our client application directly to the Facebook API is because we do not intend to take control of the user’s personal information. The server-side logic consists of back-end services responsible for extraction and filtering of deals and offers before presenting them to the user.

5.2 Evaluation of I-Shop

The usability of our implemented prototype application I-Shop is evaluated by means of the experiment described below.

5.2.1 Overview of the Experiment

I-Shop is designed to provide shoppers with smart suggestions for lowering the costs on products and improving the user experience through a personalized interface without compromising privacy. This application helps the user explore relevant deals and offers regarding products and services relevant to their current circumstances and interests. User preferences are determined in two ways:
(1) A list of products and services they have shown interest in is obtained from their Facebook profile and added to their wishlist after they have given consent.

(2) Upon arrival at a shopping center, the user is given a choice to enter preferences regarding current shopping interests.

The user is thereby presented with a list of current deals and offers based on her preferences and current location. This experiment was conducted with two participants. We installed I-Shop to their Android devices and asked them to link Facebook to I-Shop. We then asked our participants to explore shopping opportunities in their favorite malls around the city using the application. Note that I-Shop is in its prototype phase and our extended deals database contains limited information. The information recorded in our database is simulated to provide a proof-of-concept for our approach.

5.2.2 Evaluation by Participant 1
Our first participant is a working professional who has lived in Victoria, British Columbia, Canada for the past four years. She is a keen shopper and always on the lookout for bargains to make most of her time and money. Being newlywed, she is very interested in cosmetics. For this experiment she chose to visit Mayfair Mall located in Victoria. Note that some of personal information (e.g., first and last name) is randomized and generated.

Context

Source: Mobile Device
- Location: 3008 Trans-Canada Hwy (48.443672,-123.369862)
- Date: December 13, 2012
- Time: 6:15 PM
- User preferences: Shoes (Manually entered by user)
- Nearest Shopping center: Mayfair Mall

Source: Facebook Profile:
- Gender: Female
- First Language: Hindi
- Likes (Products/Services): iPhone, Mac Cosmetics

**Results**

When our first participant reached the mall, she was presented with the following offers based on interests retrieved from her Facebook profile.

**Table 2: Results based on user's wishlist (cosmetics) for Participant 1**

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Store Name</th>
<th>Deals</th>
<th>Details</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mac Cosmetics</td>
<td>Back to MAC Get FREE Lipstick</td>
<td>By returning six [6] M·A·C primary packaging containers to a M·A·C counter or online, you’ll receive a free M·A·C lipstick of your choice as our thanks to you</td>
<td>EDD (Extended Deals Database)</td>
</tr>
<tr>
<td>2</td>
<td>Sephora</td>
<td>3 philosophy bath gels for $33</td>
<td>Buy three 16 oz Philosophy Gels for a limited time value price of $33. Just enter promo code 3FOR33 at checkout.</td>
<td>EDD</td>
</tr>
<tr>
<td>3</td>
<td>Sephora</td>
<td>Free tarte pure maracuja oil deluxe sample</td>
<td>Use code MARACUJA, and get tarte pure maracuja oil FREE with purchase of C$35</td>
<td>EDD</td>
</tr>
<tr>
<td>4</td>
<td>MyCustomCase.com</td>
<td>MyCustomCase.com – Online Deal</td>
<td>C$20 for C$45 Worth of Custom Cases for the iPhone 4, 4S, or 5</td>
<td>Groupon</td>
</tr>
</tbody>
</table>

Later, she investigated deals on shoes and received the following results.
Table 3: Results based on user's selection (shoes) for Participant 1

<table>
<thead>
<tr>
<th>Store ID</th>
<th>Store Name</th>
<th>Deals</th>
<th>Details</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aldo</td>
<td>Clearance, Extra 30% off on men and women shoes</td>
<td>Limited time offer. Online only</td>
<td>EDD</td>
</tr>
<tr>
<td>2</td>
<td>Payless</td>
<td>2 pairs for just $15</td>
<td>We'll deliver your shoes and accessories to any Payless Shoe Source for free by simply choosing Ship to local Payless store for FREE at checkout.</td>
<td>EDD</td>
</tr>
<tr>
<td>3</td>
<td>Ardene</td>
<td>Free tarte pure maracuja oil deluxe sample</td>
<td>Use code MARACUJA, and get tarte pure maracuja oil FREE with purchase of C$35</td>
<td>EDD</td>
</tr>
<tr>
<td>4</td>
<td>Brida Shoes</td>
<td>Brida Shoes – Online Deal</td>
<td>Footwear and Handbags (Half Off)</td>
<td>Groupon</td>
</tr>
</tbody>
</table>

In addition, based on context retrieved from our participant's Facebook profile, she was also asked if she wishes to translate I-Shop custom messages to her native language Hindi. After approval, all the custom messages of I-Shop were translated to Hindi.

Evaluation

Efficiency: 0-1 step (i.e., selection of item)
- Based on the pre-defined user's wishlist (i.e., containing iPhone and MAC Cosmetics), deals, as shown in Table 2: Results based on user's wishlist (cosmetics) for Participant 1, were presented without any manual input (i.e., 0 Steps).
- Upon selecting Shoes from one of the available items, a list of deals for Shoes within that mall and online were presented (i.e., 1 Step, 5 seconds).

Effectiveness: High
Based on her wishlist, she showed interest in the deal on iPhone case covers and she chose to buy one.
Based on manual selection, she was not sold on the deals offered in the stores of the mall, but she was quite pleased to see the collection and half price off Brida Shoes by Groupon. She decided to look at them after she arrived at home.

User Experience: After this experiment, she was asked about her opinion on our prototype application. Based on her answers, we concluded that she was pleased with the
experience. According to her, she was driven to use I-Shop over other applications because she knew her personal information will never be exposed to a third party. On the downside, she expressed that I-Shop requires active participation from the vendors because as of now it presented deals from a few stores only. Overall she rated I-Shop at four on the scale of one to five.

5.2.3 Evaluation by Participant 2
Our second participant is a working professional, who is new to the city of Victoria and Canada. Since he is new, he is interested in exploring shopping opportunities and the city itself. For this experiment, he chose to visit Bay Center Mall in downtown Victoria.

Context
Source: Mobile Device
- Location: 3008 Trans-Canada Hwy (48.443672,-123.369862)
- Date: November 14, 2012
- Time: 5:15 PM
- User preferences: Home and Decor
- Nearest Shopping center: Hillside Mall

Source: Facebook Profile
- Gender: Male
- First Language: German
- Likes (Products/Services): Lacoste, Ray-Ban Sunglasses

Results
When our second participant reached the mall, he was presented with the following deals based on interests retrieved from his Facebook profile:
Table 4: Results based on user’s wishlist (sunglasses) for Participant 2

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Store Name</th>
<th>Deals</th>
<th>Details</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunglass Hut</td>
<td>Sale on major sunglass brands</td>
<td>Clearance sale on major brands like Dolce &amp; Gabana, Prada, Revo, DKNY, Fendi and more.</td>
<td>EDD</td>
</tr>
<tr>
<td>2</td>
<td>Zellers</td>
<td>Store closing Liquidation Sale. Up to 80% OFF</td>
<td>Offers valid while stock lasts.</td>
<td>EDD</td>
</tr>
<tr>
<td>3</td>
<td>Lacoste</td>
<td>Lacoste Designer Sunglasses – Online Deal</td>
<td>Women’s or Unisex Lacoste Sunglasses (Up to 70% Off). 18 Options Available. Free Shipping and Free Returns.</td>
<td>Groupon</td>
</tr>
</tbody>
</table>

Later, he looked up deals on *Home and Décor* and received the following results.

Table 5: Results based on user's selection (home and décor) for Participant 2

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Store Name</th>
<th>Deals</th>
<th>Details</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Quilts</td>
<td>50%-70% OFF</td>
<td>Heavy sale on Duvet covers and more PLUS free shipping</td>
<td>EDD</td>
</tr>
<tr>
<td>2</td>
<td>Zellers</td>
<td>Store closing Liquidation Sale. Up to 80% OFF</td>
<td>Offers valid while stock lasts</td>
<td>EDD</td>
</tr>
<tr>
<td>3</td>
<td>Direct Home Décor</td>
<td>Direct Home Decor – Stone Industrial</td>
<td>Custom Blinds or Interior-Design Consultation from Direct Home Decor (Up to 67% Off). Three Options Available.</td>
<td>Groupon</td>
</tr>
</tbody>
</table>

In addition, based on context retrieved from his Facebook profile, he was also asked if he wishes to translate I-Shop custom messages to his native language German. After approval, all the custom messages of I-Shop were translated to German.

**Evaluation:**

*Efficiency: 0-1 step (i.e., selection of item)*

- Based on the user’s pre-defined wishlist (i.e., containing brands such as Lacoste and Ray-Ban), the deals as shown in Table 4: Results based on user's wishlist (sunglasses) for Participant 2 were presented without any manual input (i.e., 0 steps).
- On choosing *Home Décor* from one of the available items, a list of deals on “Home Décor” within that mall and online were presented (1 Step), 6 seconds.
Effectiveness: Moderate

Based on his wishlist, this participant was presented with deals on Lacoste and Ray-Ban products. He was also given deals on sunglasses from other companies. However he chose not to buy any as during the winter he rarely wears them.

Based on his manual selection (i.e., Home and Décor), he received some deals on a wide range of products. However, he did not find anything immediately useful in those deals.

User Experience: After this experiment, he gave us some useful feedback. He was quite impressed to experience a personalized user interface as featured in I-Shop (e.g., translate messages to his native language). He was also pleased to know that his personal information was limited to his device only. Besides that, unlike some other similar applications such as Placecast [66], his smartphone does not bother him constantly with shopping alerts. I-Shop seeks permission before conveying any information.

In addition, he provided us with some useful feedback to improve I-Shop’s user interface. He said it would be better if we could provide an auto-complete text box for a user to enter an item they are interested in. At the moment he was bound to choosing from the items already present in the list. Overall he gave I-Shop a rating of three out of five.

5.3 Summary

To assess I-Shop, we conducted an experiment involving two avid shoppers. We evaluated the usability of our prototype application with three factors: efficiency, effectiveness, and user experience on a scale of one to five. The results and participant comments helped us identify both positive and negative aspects of the application. One downfall in our approach is the limited suggestions mechanism and inability to constrain search criteria. On the positive side, the participants appreciated the statelessness, internationalization, and automatic wishlist suggestion features. The end users were happy to discover that they do not have to compromise their privacy to personalize I-Shop and automatically fill their wishlist without being asked to create a profile. In the future, we may extend our database by involving vendors.
Chapter 6
Conclusions

This chapter summarizes this thesis, outlines our contributions, and sketches ideas for potential future improvements to I-Shop.

6.1 Summary
There is no doubt that the field of location based advertising (LBA) is growing enormously. It provides a path to increase sales by introducing advertisements targeted directly at nearby potential customers. However, advertisers and vendors face challenges matching current advertising technology to the changing behavioral trends of consumers to maximize sales and brand awareness. Traditional LBA applications blindly broadcast ads to mobile devices based solely on location; they fail to identify other parameters that might influence the user’s behavior. In addition, building LBA apps for smartphones across multiple platforms is an expensive and time consuming process, as each platform requires different fields of expertise.

This thesis presented the prototype application I-Shop, a context-aware LBA application for smartphones. Motivation for this research came from the identification of several limitations in the current model of mobile advertising.

(1) Tracking and storing a user’s personal information leads to privacy concerns.
(2) Limited use of context frustrates the user as the information presented is often irrelevant.
(3) Failure to understand a user’s current interests leads to spamming issues.
(4) Current smartphone application development practices are expensive and time consuming, as effective applications can only be built exclusively for their associated platforms.

The proposed generic model attempts to overcome these limitations by leveraging rich sets of context information obtained from the mobile device as well as social networking sites, all without permanently storing any user data. It utilizes the state-of-the art IBM
Worklight optimization framework to provide cross-platform support and reduce development costs, and at the same time provide a very powerful platform.

Unlike traditional advertising methods, context is our primary factor when building consumer loyalty. We try to provide personalized experiences by better understanding the user as a person. In our approach, we use Facebook’s login API to obtain existing user data. Our prototype application extracts a user’s personal information, such as name, profile picture, native language, and gender to provide a personalized user interface. Moreover, a user’s current location and interest information derived from their mobile device and Facebook profile is used to deliver the most relevant advertisements. Lastly, our approach never records any of a user’s personal information and, thus, avoids the privacy issues that keep many people from using this technology. Our approach revolves around the user—the ‘I’ in I-Shop.

Besides smart applications, this thesis also aims to discover smarter approaches in creating mobile apps. Hybrid coding, a feature supported by the IBM Worklight optimization framework, is introduced. This approach lets developers optimize their time and effort, enabling “a write once, deploy everywhere” development model that not only reduces costs but also increases efficiency by providing a combination of native and web environments.

To evaluate our prototype model qualitatively, we conducted an experiment in which two users were asked to use I-Shop in its intended capacity. After investigating their experience, it was clear that while I-Shop has room for enhancements and improvements, the application clearly demonstrated great initial potential. Users were pleased at how their information was not recorded, but still used to provide a personalized experience.

With respect to room for improvement, I-Shop did not display offers from all competing stores for a selected item. In the future, we intend to expand our database and involve active participation of vendors to provide the most up-to-date information possible. Overall, this qualitative assessment showed that our approach meets the outlined requirements and has the potential to attract a significant number of consumers.
6.2 Contributions
The section summarizes the main contributions of this thesis.

- We developed a model of a context-aware advertising application for smartphones that is smart enough to acquire relevant context, then interpret and act upon it to provide relevant information to the user at the right place and time.
- We designed and implemented the I-Shop prototype demonstrating the feasibility of our model.
- We identified a set of contextual parameters that can be leveraged to eliminate selected limitations inherent in current location-based advertising methodologies.
- We demonstrated that using Facebook’s API to extract user context provides users with a personalized experience while not violating their privacy.
- We developed a proposal for a method of creating hybrid applications for smartphones that provides a combination of web and native capabilities in a single application, eliminating the need for compromise between native and web application models.
- We used the state-of-the-art IBM Worklight infrastructure that allows developers to write software once and port to a range of devices and platforms including Android, iOS, BlackBerry, and Windows Phone using the Shell approach.
- We also identified weaknesses in the IBM Worklight infrastructure and integrated development environment.

6.3 Future Directions
6.3.1 Areas for Improvement
The proposed prototype I-Shop was designed to demonstrate the approach adopted in this research. There are a number of aspects that can be further enhanced:

- To prove the proposed model, I-Shop currently runs only on the Android platform. But since it follows Worklight’s Shell approach, it can easily be ported to the iOS, BlackBerry, and Windows Phone platforms as outlined in Chapter 3 I-Shop: A Context-Aware Shopping Advisor.
- We only focused on the application model and its user interface to provide fully functional and personalized user experiences. The aesthetics of the user interface can certainly be improved.
- To demonstrate the use case of our approach, available deals in our extended database were entered manually. This process can be enhanced by providing a web interface for vendors to directly enter information about upcoming deals and offers. Besides Groupon and our own database, more data sources could be added to provide users with a larger variety of deals.
- The recommendation mechanism could be further enhanced by setting the display order of deals based on user ranking, relevance, and proximity.

6.3.2 Public Release Potential
At this stage, I-Shop is a prototype and a proof of concept. Before a public release, the aforementioned enhancements should be made to widen the appeal of the application. For public release a public host is needed to keep the back end services running. The application is currently hosted by the Computer Science Department at University of Victoria and cannot be accessed from outside campus due to security regulations. Once these restrictions are eliminated, I-Shop can be deployed and launched via the various mobile distribution platforms including Apple’s App Store or Google Play.

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iOS</th>
<th>BlackBerry</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>App Store</strong></td>
<td>Google Play</td>
<td>Apple App Store</td>
<td>BlackBerry App World</td>
<td>Windows Phone Marketplace</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>$25 USD (one-time)</td>
<td>$99 USD per year</td>
<td>Free, with vendor account</td>
<td>Free</td>
</tr>
</tbody>
</table>
To launch a mobile app, the developer needs to register with the associated distribution platform. Table 6 illustrates the information required for registration for four major smartphone platforms.

- **Apple iOS**: Developers need to enroll in Apple's iOS Developer Program as an individual or as a company.
- **Android**: Developers need to register a Google Play account. This is a three-step process, which includes creating a developer profile, agreeing to the Developer Distribution Agreement, and paying the registration fee using Google Checkout.
- **BlackBerry**: To publish a BlackBerry application to BlackBerry App World, developers need to create a vendor account.
- **Windows Phone**: Developers need to register at Microsoft App Hub.

### 6.3.3 Commercial Potential

In November 2012, ABI Research Group conducted a survey which predicted that cumulative global mobile application revenues will exceed $30 billion by the end of 2012 [68]. Their research suggests social media and social recommendations, tactical sales promotions and discounts, branding tie-ups, and targeted in-app advertising are key aspects of monetization [67]. Clearly, there exists a market for smart, context-aware advertising applications, such as I-Shop.

If I-Shop is ever released commercially, our monetization strategy would need a web interface to gather up-to-date information on deals and offers for our extended database. These deals can then be tailored towards potential customers in the form of advertisements and sponsored listings, providing a revenue stream.
Bibliography


[38] A. Aloudat, K. Michael, L. Yan: Location-based services in emergency management—From government to citizens: Global case studies, In: P. Mendis,


[65] M. Brownlow: Smartphone statistics and market share, October 2012


Appendix A

Glossary

ABI  ABI Research is a market research and market intelligence firm
AC  Autonomic Computing
API  Application Programming Interface
ATM  Automated Teller Machine
CBS  Cellular Broadcasting Services
CSS  Cascading Style Sheet
DOM  Document Object Model
ERD  Entity Relationship Diagram
FQL  Facebook Query Language
GIS  Geographic Information System
GPS  Global Positioning System
HTML5  HyperText Markup language 5
HTTP  HyperText Transfer Protocol
IBM  International Business Machines Corporation
IDC  International Data Corporation
IDE  Integrated Development Environment
JAAS  Java Authentication and Authorization Service
JMS  Java Messaging Service
JSON  JavaScript Object Notation
LBA  Location Based Advertising
LBS  Location Based Services
OS  Operating System
RAD  Rational Application Developer
REST  Representational State transfer
SDK  Software Development Kit
SMS  Short Messaging Service
SNS  Social Networking Sites
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless local area network products based on IEEE 802.11 standards</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSRF</td>
<td>Cross-Site Request Forgery</td>
</tr>
</tbody>
</table>