Imelda Irakoze

Cloud-Based Mobile Applications

Helsinki Metropolia University of Applied Sciences
Bachelor of Engineering
Degree Programme in Information Technology
Thesis
16th May 2013
The goal of this project was to use cloud computing in order to improve the computing ability of mobile devices. Processing power, storage capacity, battery lifetime, and display size present a concern for developers when creating applications for mobile devices. In addition to this, with all the different types of smartphone operating systems on the market today, a need arises for a way to create cross-platform applications and cloud computing offers a solution to that.

Cloud computing is emerging as a dominant computing platform for providing scalable services to a global client base. Therefore, a cloud computing platform was used to develop a mobile application that supports cloud based services. Eclipse was used as a development environment for the Android application, and as Eclipse offers a Google plugin, it was also used to deploy the application to Google App Engine. Once the development environment was fully configured, it allowed the creation and deployment of a mobile application: MyNotes to the Google App Engine Cloud.

The results showed that cloud computing could be used as a backend to take some workload off the mobile device. However, MyNotes app only characterizes certain tasks such as storing, retrieving and deleting taken notes. In the future, it would be useful to test and investigate how a task that requires large computations from the mobile device or tasks that are not even possible to be performed on mobile devices, would use the cloud services to benefit mobile phone users.

| Keywords                  | mobile cloud computing, Google App Engine, mobile phones, Android, application |
Appendices
Appendix 1 Note.java
Appendix 2 persistence.xml
Appendix 3 EMF.java
Appendix 4 NoteJPA.java
Appendix 5 NoteDAO.java
Appendix 6 NoteDAOJPA.java
Appendix 7 NoteUtils.java
Appendix 8 MyNotes application code
Appendix 9 web.xml
Abbreviations and Terms

Amazon EC2: Amazon Elastic Compute Cloud is a central part of Amazon.com’s cloud computing platform: Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer applications.

API: Application Programming Interface is a software program that facilitates interaction with other software programs.

AVD: An Android Virtual Device is an emulator configuration that makes it possible to create an actual device by defining hardware and software options to be emulated by the Android Emulator.

CPU: Central Processing Unit. It is the brains of the computer where most calculations take place.

DAO: Data Access Object is an object that provides an abstract interface to some type of database or other persistence mechanism.

EMF: Eclipse Modeling Framework is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model.

GFS: Google File System is used by Google to organize and manipulate huge files and also to allow application developers the research and development resources they require.

GPS: The Global Positioning System is a navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth.

Google Big Table: is a data storage facility for Google. Bigtable is a distributed, persistent, high-performance, multidimensional sorted map. Bigtable is not a relational database.

HTML5: HTML5 is a markup language for structuring and presenting content for the World Wide Web and a core technology of the Internet.
IDE: An Integrated Development Environment is a software application that offers comprehensive resources to computer programmers for software development. An IDE consists of a source code editor, building tools and a debugger.

iOS: iPhone Operating System is a mobile operating system developed and distributed by Apple Inc.

IaaS: Infrastructure as a Service is a service model that provides storage, infrastructure and other hardware assets as resources that clients can provision.

Jetty: Is a Java-based HTTP server and a Web container. A web container is the component of a web server that interacts with Java servlets. A web container is responsible for managing the lifecycle of servlets, mapping an URL to a particular servlet and ensuring that the URL requester has the correct access rights.

JDO: Java Data Objects is a standard way to access persistent data in databases, using plain old Java objects (POJO) to represent persistent data.

JPA: Java Persistence API is a standard interface for storing objects containing data into a relational database.

JVM: A Java Virtual Machine is an implementation that interprets compiled Java binary code for a computer's processor so that it can perform a Java program's instructions.

MySQL RDBMS: is open source Relational Database Management System that runs as a server providing multi-user access to a number of databases.

OS: The Operating System is a base infrastructure software component of a computerized system. It controls all basic operations of the computer.

PaaS: Platform as a Service is a concept that describes a computing platform that is rented or delivered as an integrated solution, solution stack or service through an Internet connection.

SaaS: Software as a Service is a model for the distribution of software where customers access software over the Internet.
Smartphone: A smartphone is a mobile phone with highly advanced features.

SDK: A Software Development Kit is a programming package that enables a programmer to develop applications for a specific platform. It includes APIs, programming tools, and documentation.

SQL: Structured Query Language is a standard language for accessing and manipulating databases.

TTS: Text-to-Speech enables the Android device to speak text of different languages.

URL: A Uniform Resource Locator is the address of a resource on the Internet.

Wi-Fi: Wi-Fi is a wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections.

XAMPP: Cross Apache MySQL PHP Perl is a free and open source cross-platform web server solution stack package, consisting mostly of Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

XML: Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both self-descriptive and also machine-readable.

3G: short for third Generation, is the third generation of mobile telecommunications technology.

4G: 4G is the fourth generation of mobile phone mobile communication technology standards.
1 Introduction

Mobile devices have become an essential part of our daily lives. From being a device mainly used for phone calls and text messages, the mobile phone of today has become a multifunctional device. Smartphone sales now exceed desktop system sales. According to the mobile factbook, mobile subscribers worldwide were 5.3 billion at the end of 2010, but they are set to reach 7.4 billion by the end of 2015 [2]. Improved networks, attractive tariff plans, affordable handsets and the availability of enhanced data services such as mobile apps, mobile payments and high-speed mobile Internet have influenced this remarkable growth. As mobile device popularity grows, end-user demands to run heavier applications are also increasing.

The need to maintain the advantages mobile devices have over desktops and laptops in weight, size, and device autonomy will always dictate certain limits on processing power, storage capacity, battery lifetime, and display size [1]. Developers must redesign standard desktop applications to run on mobile hardware platforms, which might require reducing functionality, while more demanding applications need specific hardware resources that are unlikely to be available on mobile devices.

Over the past few years, smartphones have proven that they are immensely capable; they are now almost as powerful as a desktop or laptop computer. However, there are applications that require much more processing power and Cloud computing can enhance the computing ability available to mobile users by supporting applications requiring large storage capacity, graphical hardware, such as 3D virtual environments or other cloud services unavailable on mobile devices. Mobile cloud computing offers an approach to meet users’ increasing functionality demands. However, short battery lifetime, varying wireless channel conditions, and interaction latency present major challenges for some cloud applications on mobile devices.

The goal of this project is to gain a better understanding of cloud-based mobile applications. Cloud computing concepts and core ideas are discussed, as well as the existing problems and developments in mobile cloud applications. To further understand the topic, a mobile application will be developed and deployed to the cloud.
2 Cloud Computing

2.1 Definition of Cloud Computing

Nowadays it is impossible to read a technology journal or blog without coming across the term cloud computing. While some might think that cloud computing is just a new buzzword, something companies use to sell services, cloud computing is transforming the way we deploy technology. The cloud is often used in a very general way and labeled on products that are not necessarily cloud computing services, but this paper provides a perspective on cloud computing and sheds light on the sometimes ambiguous understanding of cloud computing.

Cloud computing is not just a service being offered from a remote data center. It is a set of approaches that can help organizations quickly, effectively add and subtract resources in almost real time. Cloud computing provides the means through which resources such as computing power, computing infrastructure and applications can be delivered to users as a service wherever and whenever they need over the Internet. [3] Cloud services include the delivery of software, infrastructure, and storage over the Internet based on user demand. Mell and Grance from the U.S. National Institute Standards and Technology defined cloud computing as “a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”[46]. Plummer, Bittman, Austin, Clearley, and Smith (2009) explained it as “a style of computing where massively scalable IT-enabled capabilities are delivered as a service to external customers using Internet technologies” [12].

The 451 Group defined cloud computing as “an IT as a service, delivered by IT resources that are independent of location [13] and Buyya, Yeo and Venugopal (2008) define the cloud as "a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers”[14,601].
After examining the definitions given of cloud computing it helps us clarify the term and what it involves; briefly cloud computing is a way of delivering computing services such as software, servers or storage over the Internet in a self-service manner. Instead of having to install, maintain and manage these resources, one only needs to access and use them through a web browser or a specifically designed user interface.

Cloud computing can be used to overcome the limitations of data centers. An enterprise data center is where servers and storage are located, operated and managed. A functional data center requires a lot of power, a lot of space, cooling, maintenance and so on. Most of human activities such as energy, lighting, telecommunications, Internet, transport, urban traffic, banks, security systems, public health and entertainment are controlled by data centers. People rely on the functioning and availability of one or multiple data centers. The process of adding and releasing resources in the traditional data center cannot be done in an automated or self-service manner, but in the cloud, users can request extra resources on demand and also release them when they are no longer needed. The fact that the cloud can easily expand and contract is one of the main characteristics that attracts users and businesses to the cloud.

Furthermore, as enterprises grow and there is a need for more resources, IT departments usually add hardware to the data center and buy new software, which makes the data center even more large and complicated. Managing a big data center that is still expanding is stressful for IT management, thus the introduction of technology advancements such as virtualization. Even though these advancements in technology have enabled much more efficiency and cost effectiveness, companies are still overwhelmed with a lack of ability to satisfy customers’ needs. The public cloud enables companies to make use of external resources to improve their ability to offer services requested by users without investing in new infrastructure, training new personnel, or licensing new software. With cloud computing, there are no servers to maintain, which gives companies time to focus on other tasks; hence improving productivity. The public cloud involves end users that benefit from cloud services without knowing the underlying technology and cloud service providers that are responsible for the IT assets and maintenance.
2.2 Cloud Computing History

Cloud computing has been possible through development in a variety of areas [4;5]. As computer hardware evolved, so did software, as communication networks developed so did the protocols for how computers communicate. The communication rules and standards in turn affected the evolution of Internet software that made cloud computing possible [6].

According to Carr, what is occurring nowadays is very similar to what happened during the industrial era. During the industrial era many industries had to provide their own electricity by wind or water mills to power their machines. As electricity through power lines became cheaper, more available and more reliable, there was no need for the industries to produce their own energy. Carr says that the organizations of today face the same shift as the ones during the industrial era, except today the shift is toward cloud computing. [7]

Toffler states that a civilization goes through different waves of development: the first wave was the agricultural societies, the second one was the industrial age and what we are now facing is the third wave; the information age [9]. Development in various areas have led to the development of cloud computing. Some areas have affected the growth more than others, for example virtualization, utility computing, outsourcing and grid computing, are areas that have had a great impact on cloud computing.

2.2.1 Virtualization

IBM introduced virtualization in the early 1960s [3]. Virtualization makes it possible to run several operating systems on one server simultaneously, but separates them as if they were on their own server. The difficulty comes from the fact that today’s computers are designed to run just one operating system and application at a time. Virtualization allows multiple operating systems and applications to share a single hardware host. Each virtual machine is isolated from the others by the hypervisor, and uses as much of the host’s computing resources as it requires. A hypervisor monitors the virtual machine in a way that each operating system appears to have the host’s processor, memory, and other resources all to itself.
The hypervisor controls the host processor and resources, allocating what is needed to each operating system, and making sure that the guest operating systems (virtual machines) cannot disrupt each other. The difference between virtualization and cloud computing is that virtualization is part of a physical infrastructure, while cloud computing is a service. There is also a difference in costs: using virtualization requires upfront costs but with cloud computing the charges are based on how much resources are used.

2.2.2 Utility Computing

“Utility computing can be defined as the provision of computational and storage resources as a metered service, similar to those provided by a traditional public utility company” [6]. The idea of utility computing is to provide computing resources like how electricity, telephone or water is provided. We pay for the amount that we use and it should be available to us.

The main benefit of utility computing is better economics. Corporate data centers are underutilized, with resources such as servers often idle 85 percent of the time. Utility computing allows companies to only pay for the computing resources they need, when they need them. Utility computing differs from cloud computing by the fact that it relates to the business model in which application infrastructure resources are delivered, whereas cloud computing relates to the way we design, build, deploy and run applications that operate in a virtualized environment, and offering the ability to dynamically grow, reduce and self-services.

2.2.3 Outsourcing

Outsourcing is the act of one company contracting with another company to provide some services. Often the company itself could perform the tasks that are outsourced, but in many cases there are financial advantages that come from outsourcing. In comparison to cloud computing there are similarities but also some differences. Companies can outsource parts, or their whole IT department, on companies specialized on that particular field. For example a company that outsources the setup, maintenance and storage of their servers so that they do not have to have them onsite, on their company compound. This is similar to PaaS or IaaS where cloud computing vendors take care of the platform and/or infrastructure.
There are many similarities but the differences lie in the quickness of providing the services and agreements from the outsourcing company or the cloud provider. Unlike traditional outsourcing that requires lengthy contracts that usually just carry on as long as the contracts agree on, cloud computing offers a predefined solution that matches the need of the customer’s application [6]. There is usually no initial cost, and the customer only pays for what is being used and nothing more. There are also some differences in the level of management, security and support when comparing cloud services and traditional outsourcing.

2.2.4 Grid Computing

The term “Grid computing” originated in the 1990s and refers to the idea of making computing accessible in similar manner to how a power grid works [4]. “Grid computing is a form of distributed computing that implements a virtual supercomputer made up of a cluster of networked or Internetworked computers acting in unison to perform very large tasks” [6].

Many cloud service providers offers services similar to grid computing by the pay-per-use model and perceived unlimited computing resources. However, cloud computing should be viewed as a step away from the grid utility model [6]. The fields overlap each other on several points, but the main difference between the two of them is how data is processed. In grid computing the user usually makes few but very large request. Only a few of these requests can be processed at any given time and others might be queued. However, cloud computing users do several small allocation requests, where allocations happen in real time [4].

2.3 Cloud Computing Features

Overall, the cloud embodies the following key features:

- Resource pooling and Elasticity
- Self-service and automatic services
- Access over the Internet
- Billing in a pay-per-use model

Each of these characteristics is described in more detail in the following sections.
2.3.1 Resource Pooling and Elasticity

Resource pooling is the ability to scale up and down to serve multiple customers using a multi-tenant model with different physical and virtual resources dynamically assigned and reassigned according to demand. Often, the service provider cannot predict how customers will use the service. Some customers might use the service a few times during their highest seasons, while others might use it as a primary development platform for all of its applications. Therefore, the service needs to be available all the time and it has to be designed to scale upward for high periods of demand and downward for lighter ones. The service also needs to scale when additional users are added and when the application requirements change. [3]

2.3.2 Self-service and Automatic Services

Cloud Computing allows customers to get cloud services easily without going through a long process. The customer simply requests an amount of computing, storage, software, process, or other resources from the service provider. This is an advantage that cloud computing offers compared with the process one has to go through when requesting new services from a typical data center. Before implementing a new application, the IT department has to submit a request to the data center for additional computing hardware, software, services, or process resources. The data center evaluates all requests from various departments and assesses the availability of existing resources versus the need to purchase new hardware. After new hardware is purchased, the data center is configured for the new application. This proves to be a long and complicated process that can be made easier by using cloud services. [3]

2.3.3 Access over the Internet

The access of services over the Internet allows convenience. Access over the Internet means that resources hosted in the cloud are available for access from a wide range of devices and from several locations that offer online access [3]. Users are able to access data and services wherever and whenever they need, from the home computer, tablets, or smartphones. Usually, this was done through a browser, to avoid the need to install local software.
However, cloud-based applications are now available in order to access data anytime it is needed. As further discussed in section 2.3.4, cloud computing has different deployment models: private, public and hybrid clouds. In a private cloud, secure data is accessed only by company employees within a company’s own firewall. The company operates its own infrastructure, including a data center full of servers. Public cloud computing is when companies use an outside company to host servers or other cloud services that the company accesses for its employees. Access over the Internet might cause some security issues in a private cloud, but, as more employees use smartphones, tablets to access company resources or may want to work from their homes, a need for a network access over the Internet arises which may cause the company to adopt a hybrid cloud that combines both a private and a public cloud.

2.3.4 Billing in a Pay-Per-Use Model

Pay only for the services used, and no more. Rather than paying a 100 percent for servers that are only used 20 percent of the time, one only pays for the exact number of resources used. The ideal cloud providers charge usage in terms that everyday people; not just IT systems administrators, understand. A cloud environment needs a built-in service that bills customers. Of course, to calculate that bill, usage has to be tracked. [3] For example measuring the storage, bandwidth, and computing resources consumed, and charging per stored gigabytes, transferred bytes, used computing hours, number of active user accounts per month or performed transactions.

A pay-per-use model involves different payment mechanisms, such as subscription based payment, reservation based payment, consumption based and so on. In subscription based payment, the user has monthly or yearly fees to access the service. A pay-per-reservation method is when the user pays for the duration of the service. How much of the infrastructure used from the moment the service started is not necessary, the user will be charged according to the time. In consumption based, the user’s consumption is measured and charged according to the amount of memory, CPU cycles, disk space, and network traffic. These examples show that there are different approaches to the pay-per-use model, but they are all based on the amount of resources used or the period the service was available.
2.4 Cloud Computing Deployment Models

A cloud deployment model defines where the physical servers are deployed and who manages them. Cloud computing deployment models are:

- Public cloud
- Private cloud
- Community cloud
- Hybrid cloud, which combine both public and private

In the public cloud, the infrastructure is designed to make the services available to public users on the Internet. For the private cloud the infrastructure is configured exclusively for a private user, meaning an enterprise or organization where the services can only be accessed locally, and the organization’s IT department manages it. The infrastructure for the community cloud is shared by several organizations and supports a specific community that has common concerns (e.g. security requirements). The organization or a third party may manage it. As for the hybrid cloud the infrastructure is comprised of two or more clouds (private, community, or public) that remain unique entities but are communicating with one another by standardized technologies that enable data and application portability [46].

2.5 Cloud Computing Service Models

The three cloud service delivery models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These services are classified into three models to help us understand them; they are tasks that have been put together in order to be delivered to customers whenever they need them. The Infrastructure as a Service layer offers hardware, storage, servers, data center space and network components that developers and IT organizations use to build and deliver solutions. The Platform as a Service layer offers development environments, which software developers can use to create fully functional products or services. [3]

The Software as a Service layer offers software over the Internet. The customer accesses those services with defined interfaces. These interfaces are all that the user ever comes in contact with.
For example when watching a movie via Netflix the customers only sees the screen that enables selecting and watching the movie, they never see the underlying infrastructure and what happens behind the scenes to allow Netflix to deliver so many movies to many people. As illustrated in figure 1, the various types of cloud services are included into three distinct models.

![Figure 1. Cloud computing service models](image)

In cloud computing the underlying infrastructure that provides the service may be very complex, but the user does not need to understand this infrastructure in order to use it. The following sections will help us explore and understand cloud computing service models.

2.5.1 Infrastructure as a Service

Infrastructure as a Service (IaaS) is the delivery of computer hardware (servers, networking technology, storage, and data center space) as a service. It may also include the delivery of operating systems and virtualization technology to manage the resources. Instead of buying and installing computing resources in their own data center, customers rent them. An IaaS provider is responsible for operating and maintaining the equipment it provides for a customer. Clients pay on a per-use basis. One of the characteristics of IaaS includes dynamic scaling to ensure more resources will be automatically given to the client in case they need them.
Also, the service involves an agreed-upon service level in terms of availability and response to demand. For example, it might be stated that the resources will be available 99.999 percent of the time and that more resources will be provided dynamically if greater than 90 percent of any given resource is being used. [3] An example of an IaaS is Amazon's Elastic Compute Cloud (Amazon EC2). It provides a web interface that allows customers to access virtual machines. The use of the term elastic in the naming of Amazon’s Infrastructure as a Service refers to the ability that EC2 users have to easily increase or decrease the infrastructure resources assigned to meet their needs. The user needs to initiate a request; the service provided is not dynamically scalable.

2.5.2 Platform as a Service

Platform as a Service (PaaS) is a concept that describes a computing platform that is rented or delivered as a solution stack, which is an integrated set of software that provides everything a developer needs to build an application. The PaaS service delivery model allows a customer to rent virtualized servers and associated services used to run existing applications, or to design, develop, test, deploy and host applications.

Recently web-hosting companies have been offering software stacks for developing web sites. PaaS can be considered an advancement of web hosting because it provides also a lifecycle management which is the process of managing the entire lifecycle of a product. It also involves all the software development stages from planning and design, to building, deployment, testing and maintenance.[3] The main advantage of PaaS is not having to worry about managing and maintaining the infrastructure, and only focus on developing the product. PaaS is also delivered with dynamic scaling; it can automatically scale up or down. An example of Platform as a Service is the Google App Engine.

2.5.3 Software as a Service

Software as a Service (SaaS) is a model for the distribution of software where customers access software over the Internet. In SaaS, a service provider hosts the application at its data center and a customer accesses it via a standard web browser [46,1-3].
SaaS has its roots in Application Service Providers (ASPs), which used to host and manage business applications. SaaS extends the idea of ASPs; they require installation of software on users’ personal computers, but SaaS solutions rely on the web and only require an Internet browser for users to access services. Also, while most ASPs maintained a separate instance of the application for each business, SaaS solutions normally utilize a multi-tenant architecture, in which the application serves multiple businesses and users, and partitions its data accordingly. The price of the software is on a per-use basis and businesses are able to reduce capital expenditures. Furthermore, before acquiring new software, businesses can test them first on a rental basis and if they find them appropriate, they can purchase them. Examples of Software as a Service are Microsoft Office 365 and Google drive.

2.6 Advantages of Cloud Services

There are several benefits with cloud computing that companies can use to reduce costs while providing a high level of service to customers. This section will show how cloud computing can benefit an organization.

2.6.1 Improved Business Agility

In the ever-changing market, businesses need to be able to adapt rapidly and cost efficiently to changes in the business environment. Cloud computing offers a way to save time and money by providing the ability to add new infrastructure quickly and in a self-service manner. Managing the internal information systems is not the core competence for most companies [18]. Therefore, by using cloud services, IT management will be able to focus on supporting the corporate business values, while the cloud provider takes care of the IT needs [19].

2.6.2 Reduced Costs

With the introduction of the cloud, companies can test a new application or develop a new application without first investing in hardware, software, and networking, and if they like it they would go ahead and purchase it. A need might arise to increase storage or buy new software for various departments, but there is not enough money to buy all those services at once. Cloud service vendors might rent storage on a per-gigabyte basis.
Companies are often confronted with the need to improve the functionality of IT while reducing costs. By adopting the cloud computing pay-as-you-go model, where one only pays for the amount of resources they use or respectively for the time the service is accessible, companies are able to avoid a large initial investment and instead pay for the functionality as an operating cost.

2.6.3 Elasticity and Scalability

Computing resources are dynamically assigned, released, and reassigned according to consumer demand. Elasticity means that the platform can handle sudden, unexpected, and large loads. This could be due to an event happening that results in a vast but short influx of users on the system. Scalability is a planned level of capacity that is able to scale up or down in a quick and easy manner when more or less resources are needed. In cloud computing resource allocation can get bigger or smaller depending on demand. For example, an application can scale when adding users and when application requirements change.

2.6.4 Rapid Application Development

Using cloud computing to build, test, and deploy applications, reduces the overall development time, due to the cloud platform's ability to simplify the development process and the ability to quickly get the development resources online. Cloud-based development platforms in PaaS and IaaS clouds, such as Google, Amazon Web Services, Microsoft, and Salesforce.com offer developers the ability to self-provision development and testing environments without having to wait for hardware and software to be installed in the data center. It allows them to quickly get applications into production and to scale those applications as required. This enables cost savings and efficiency.

2.7 Disadvantages of Cloud Services

Companies require an ideal performance, a perfect implementation and a 100 percent uptime in order to satisfy customers. They also want to be able to get new infrastructure quickly, but have limited budgets. While cloud computing can offer all that, it comes with a few flaws. This section introduces us to cloud computing disadvantages.
2.7.1 Security

There is a concern in larger organizations towards turning over their operations and data to a cloud-based service provider [20]. A recent study done by the International Data Corporation (IDC), shows that almost 75 percent of Chief Information Officers and IT executives are concerned about the security when using cloud services [18]. Before using cloud services, companies need the right level of security to make sure that another company cannot access the information or be maliciously accessed by a hacker.

2.7.2 Vendor Lock-in

When using Platform as a Service, customers may find it difficult to move their applications to another development environment without rewriting them. When in need to switch to another PaaS vendor, rewriting the applications might cost companies plenty of money. This drawback has opened up a new approach: Open Platform as a Service. [3] This is the same service as Platform as a Service, but customers are able to choose through a variety of development software, there are no limitations and it prevents customers from being locked in. [3]

2.7.3 Misuse of Data

There is a risk that data could be permanently lost by a cloud computing service provider due to various reasons such as technical errors or physical disasters like fire. Also, a crooked employee that has access to the data might misuse it or the data might be compromised by external parties. Even though these issues are not likely to occur, because the providers usually have back-up and proper security techniques, it is important for a company to consider how to address data loss or misuse in its agreement with the provider.

2.7.4 Lack of Maturity

Cloud computing is still in its new phase and users may not fully understand how to utilize the capabilities of the concept [14]. Users might ask themselves questions like what happens to their data when the cloud provider is no longer there.
Currently, there is no way for a cloud storage service provider to directly transfer customer data to another provider. If a service goes down, the hosting company must return the data to its customer, who then must find another provider. In addition, there are no rules regarding data removal. When a customer asks a cloud vendor to delete some of their data, it is not done right away. Cloud service providers use a "garbage collection" method for deleting old data. The data to be erased is marked first, then the actual deletion or overwrite process takes place at a later date, sometimes months later. [45]

3 Mobile Cloud Computing

Mobile cloud computing is the availability of cloud computing services in a mobile ecosystem, in other words using Cloud Computing principles to deliver applications and services for mobile devices [22]. Developments in mobile hardware and software have enabled users to perform tasks that were once only possible on personal computers and other devices like digital cameras and GPS navigation systems. Mobile users are now connected to the Internet, they can capture and manage photos and videos, play music and movies, and play complex games. However, the increasing number of mobile applications requires more resources in terms of storage and processing capability. Mobile devices compared to desktop computers, have less computing power, less storage capacity and battery limits. Demanding applications such as video streaming and mobile games need more resources on mobile devices for a better user experience. Migrating computing and major data processing tasks to the cloud can fill the gap between resource demand and supply in mobile devices.

Definitions of mobile cloud computing can be classified into two categories. The first one denotes carrying out data storage and processing outside mobile devices [25]. Mobile devices tasks are reduced because the storage and computing processes take place in the cloud. The second category refers to mobile cloud computing as an extension of cloud computing in which foundation hardware consists at least partly of mobile devices. This definition acknowledges an opportunity to harness the collective sensing, storage, and computational capabilities of multiple networked phones to create a distributed infrastructure that can support new applications. By using the combined data and computational abilities of an entire network of smartphones, useful results for clients both outside and within the mobile network can be generated.
This interface and the underlying hardware would create a mobile-cloud upon which certain mobile phone tasks could be performed. [25] This section introduces the concepts of mobile cloud computing, discussing its challenges and solutions. Figure 2 illustrates a general architecture of mobile cloud computing. In figure 2, mobile devices are connected to the mobile networks via base stations (e.g., base transceiver station (BTS), access point, or satellite) that establish and control the connections between the networks and mobile devices. Mobile users’ requests and information (e.g., ID and location) are transmitted to the central processors that are connected to servers providing mobile network services.

As shown in figure 2, mobile cloud computing can be simply divided into cloud computing and mobile computing (mobile devices, mobile applications, the infrastructure of mobile networks and protocols). Those mobile devices connect to a base station by 3G, 4G, WIFI, or GPRS.
Mobile network operators provide the necessary services to mobile users such as provisioning, billing and AAA (Authentication, Authorization, and Accounting) based on the home agent (HA) and subscribers’ data stored in databases. After that, the subscribers’ requests are delivered to a cloud through the Internet. In the cloud, cloud controllers process the requests to provide mobile users with the corresponding cloud services.

3.1 Challenges

3.1.1 Resource Limitations

The main issue with the mobile cloud is the resource limitations of mobile devices. Compared to desktop computers, they have less memory, less compute power, and battery capacity limits. The mobile cloud is often viewed as SaaS, meaning that computation and data handling are usually performed in the cloud. Smartphones often access the cloud through web browsers or thin clients. [23] Reduced battery lifetime is a fundamental challenge for mobile devices. Mobile devices are less powerful and use a battery, whose capacity is limiting and prevent users from completely relying on their mobile device. It is therefore important to maximize battery life through the careful partitioning of application functions across servers and devices.

3.1.2 Latency and Bandwidth

Latency and bandwidth affect the mobile cloud as well. Wi-Fi improves latency but may decrease bandwidth when many mobile devices are present. Wireless connectivity is characterized by variable data rates and intermittent connectivity due to gaps in coverage. The dynamic nature of application throughput demands, subscriber mobility and uncontrollable factors such as weather, can cause bandwidth capacity and coverage to vary. Bandwidth for 3G cellular may further be limited by cell tower bandwidth in some areas. Similarly, connectivity may be irregular. [23] Internet service providers are implementing 4G networks to help them meet the growing end-user demand for more bandwidth, higher security and faster connectivity on the move. [26]
3.1.3 Security Issues

Security issues increase with mobile devices. It is easier to lose a mobile device and in case that device contains sensitive data just downloaded from the cloud, it could lead to an information leakage and data loss. Cloud computing users prove their identities with digital credentials, typically passwords and digital certificates. If an attacker could fake or steal these credentials, the cloud computing system will suffer from spoofing attacks. Mobile devices have less computing power to execute sophisticated security algorithms; therefore mobile cloud computing is more prone to attacks from hackers. In addition, it is difficult to enforce a standardized credential protection mechanism due to the variety of mobile devices. [30]

3.2 Solutions

Various solutions have been proposed to challenges often faced by mobile cloud computing. In this section, some of the solutions are reviewed.

3.2.1 Offloading Mobile Applications

Offloading mobile applications to the cloud is a way to save on device energy consumption because it reduces the amount of local processing. However, it is not possible to completely delegate the execution of all applications in the cloud. Application architects need to think about partitioning application functionality that can be offloaded on cloud versus executed on the mobile device. For example some non-display applications like virus scanning are more suited for being offloaded to the cloud. To achieve seamless and transparent migration and offloading, the application should be partitioned, meaning dividing the complex workload into lighter components that can be processed simultaneously. [23]

3.2.2 4G Technology

One of the biggest enablers for network reliability in mobile cloud computing will be the full implementation of 4G Technology, which will help with issues of latency and bandwidth. HTML5 also allows specification of offline support, which makes local storage possible, helping with connectivity interruptions.
One example of an HTML5 benefit is the ability to watch a video without a plug-in like Adobe Flash or Microsoft Silverlight. HTML5 features improvements in forms specifications that benefit mobile applications. [23]

3.2.3 Embedded Hypervisor

An embedded hypervisor will enable cross-platform applications. The hypervisor allows a web application to run on any smart phone without being aware of the underlying architecture. Mobile platforms require the hypervisor to be built in. For example, the Motorola Atrix has an embedded hypervisor that allows it to run a wider range of applications, not just those developed specifically for it. [23]

3.2.4 Security

As mentioned in section 3.1.3, mobile phones get lost easily. Therefore, there should be a way to prevent data misuse from lost or stolen devices. One way is the ability to wipe off mobile devices remotely. Some mobile manufacturers and wireless carriers provide this feature. The risk of privacy exposure and identity theft can be reduced by implementing improved protection measures for sharing data in interconnected systems, implementing monitoring capabilities and protocols, and by educating users about proper social media safe-surfing [31]. One added security feature is lightweight virus and using firewall clients on mobile devices.

4 Cloud-based Mobile Applications

With the rapid adoption of smartphones and tablets, companies need to add mobile applications to their service portfolios in order to reach more customers and provide services anywhere, anytime. Apps are actual applications that are downloaded and installed on mobile devices, rather than being accessed within a browser. In order to download apps, users visit mobile application stores like Apple’s App Store, Android Market, or Blackberry App World depending on the device’s operating system. The app may pull content and data from the Internet, in a similar way to a website, or it may download the content so that it can be accessed without an Internet connection.
Mobile apps allow content to be available offline and provide access to the phones’ camera or phone resources to store information or process complex data. Functions can also be performed without an Internet connection [42]. Examples of mobile apps are Angry Birds, Facebook and Gmail.

Mobile apps enable interactive games, and when developing productive apps like EverNote or SportsTracker that involve users’ personal and daily usage, apps are a useful way to satisfy users’ needs. In addition, apps allow users to access their banking systems, which would not be safe to do through a usual web browser because of the lack of security systems in mobile phones. Mobile cloud computing provides many advantages for application developers as well as their end-users.

With cloud-based applications, mobile users do not need high-end hardware and infrastructure to run or maintain mobile apps. Mobile app development requires hardware and software resources, and for organizations that do not have enough funds to get started, cloud computing comes in to provide needed resources in a pay-as-you-go model. Cloud computing services are easy to use, they are scalable, efficient and offer a pay-per-use pricing. On-demand services and the availability of limitless processing power and storage provided by the cloud allow developers to reach high levels of mobile app functionality. [43]

When using cloud computing services, developers do not need to be concerned about deployment or the operational maintenance of the infrastructure to power their applications, they can focus on developing the main task at hand, while other functionalities are being taken care of. Cloud computing offers the performance and flexibility that mobile app developers require. With the introduction of cloud computing, the smartphone and app markets began to grow at a notable rate with more innovative and useful apps developed more quickly and cost effectively. This chapter discusses the benefits of cloud-based applications for developers.

4.1 Cost Advantages

Developers do not need to invest heavily in building infrastructure and resources. Cloud computing provides instant access to scalable mobile application tools for building mobile and tablet apps. Cross-platform app development also helps reduce costs. Developers can now build an app once, test it and deploy it across multiple platforms.
Building once and deploying to many devices considerably decreases the cost of developing apps. In addition, deploying apps to app stores and web sites is much easier. For example, developers can avoid device manufacturers or carrier app stores to distribute their apps, and publish them on their own private channels. [43]

4.2 Increased Developer Productivity

Developers can implement their ideas without being concerned about the infrastructure or the capacity of the surrounding services. In a usual on-premise infrastructure, developers need to acquire the virtual machine (VM) capacity for computing tasks, and database capacity for persistence, and they must build a target infrastructure environment even before implementing simple programs. [34] Cloud allows developers to acquire compute, storage, network infrastructure, and managed services easily and quickly. Therefore, developers will be motivated to try new ideas that may drive organizations into new market places.

4.3 Secure Applications with Pre-defined Security Frameworks

Developers do not have to create their own code to allow industry-standard authentication and authorization techniques in their web applications. For instance, if an application needs to make use of several OpenID providers (Microsoft Live, Google ID, Facebook, or Twitter to name a few) on the Internet, the developer must manually write forking code to understand multiple tokens, parse them to a canonical structure, and apply authorization rules before the user is able to access an application functionality. [34] Some service providers simplify this whole process through simple settings. All the required implementation is already configured into a format that the application understands. Such services also free the developer from understanding the complicated details of OpenID protocols, and the token technicalities of each OpenID provider implementation. [34]

4.4 New Platform Capabilities

Cloud vendors usually bring new capabilities to market constantly at an increased rate compared to that of upgrades to on-premise software packages and operating systems. This is due to the fact that cloud services run on a standardized hardware and software platform inside the providers’ own data centers. This allows the deployment of features in a controlled manner with predictable impact to the deployed customer applications. This accelerated feature delivery allows a developer to take full advantage of the vendor’s investments that help their enterprises develop new solutions. [34]
4.5 Up-to-date Application Platforms

Cloud providers easily deliver new releases of technologies. As soon as there are updated versions, developers can start working with latest versions. They do not have to be slowed down by on-premise IT deployment latencies anymore. In addition, they are able to prevent the problems of being slowed down by outdated infrastructure and maintenance delays. A developer can integrate the latest development trends into the application as the recent frameworks and services usually perform with increasingly smaller amounts of code. This is a benefit to the companies they work for, as well as for the developer himself as this increases his skills sets. [34]

4.6 Improved Reusability of Services

Because of financial restrictions and delivery plans, developers usually focus on meeting the needs of the current project when they create and deploy services. Therefore, scalability issues may occur once consumers are using the services across the enterprise. With a cloud-based service, the developer does not need to worry about scalability; the required service will scale up or down depending on the demand.

4.7 Use of Existing skills for Cloud Applications

In many cases, existing core skill sets transfer directly to cloud technologies. The need for design skills remains. The critical success factor of the broader adoption of a cloud platform is the developer ecosystem. Most cloud platforms allow the reuse of the existing skills either in the Java or .NET space. The cloud platforms are highly compatible with their on-premise alternatives, making applications highly portable across deployments. Most of the server-based web applications and web services can be ported with minimal or no changes. [44]

Briefly, cloud computing enables developers to build highly scalable, available, reliable, and high-performance platform independent applications, with a shorter time to the market. Cloud-based mobile application market is expanding at an exponential rate and it is a market changer and a new industrial revolution.
5 Building a Cloud Aware Mobile Phone Application

Yang et al. (2008) used a mobile phone application to investigate if it was possible to efficiently offload application functions using the cloud [10]. This approach was also chosen in this paper, but the idea is to build a mobile phone application, deploy it to Google App Engine and store applications data into the datastore. This task is used as a proof of concept to demonstrate how applications can use cloud computing to take some workload off locally from the mobile device to the cloud. This section shows the path to building MyNotes, an application that takes notes and displays them. The first step is to build an application that takes notes and displays them to a web interface and to a mobile applications’ screen on the smartphone.

5.1 Choosing a Cloud Vendor

Choosing a cloud computing vendor can be difficult because there are hundreds of options, one of the major ones are Amazon, Microsoft and Google. The types of services they offer are different; Amazon EC2 offers the possibility to customize the cloud server in many ways, for example by offering which operating systems and which software that should be running on the virtual server. The advantage of EC2 is that there is a large possibility to build and customize the server, but it also requires some work to set it up.

Microsoft and Google offer the leading Platform-as-a-Service solutions, thus a special attention was given to these two. Azure and App Engine has been compared before, in [36] Smith compares the platforms by developing the same application on both platforms and concludes that App Engine provides an easier learning curve but that Azure is a good choice for existing .NET developers. Singh (2009) compares the platforms in several categories. Azure comes first in language support, application types, and customized solutions. App Engine is the best in Development and price, but both the platforms offer good services in scalability and storage. These comparisons are a helpful way to decide which cloud vendor is suitable for a specific application, but it is not wise to completely rely on them, because platforms are constantly being updated with new features. [37]
Microsoft Azure only offered the possibility to develop applications in .net or PHP, which moved the focus to Google App engine as it offers an option of developing in Java. In addition, App Engine offered their services for free. Because the workload on the cloud servers is expected to be quite low and because Java servlets were used, the choice of cloud vendor fell on Google App Engine. Google products are commonly used in everyday life and the reliability, performance and security that they demonstrate acted in favor of Google App Engine. Google App Engine is a cloud computing PaaS. It enables developers to build applications on Google infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale. As the number of requests and data storage of the application increases, App engine allocates resources to deal with the additional demand. With App Engine, there are no servers to maintain: One only needs to upload the application, and it is ready to serve users. [35]

The following section explores the features of Google App Engine.

Google App Engine provides a **fully integrated development environment**, there is no assembly required, which makes it easy to get started. App Engine is a complete development stack that allows developers to use familiar technologies to build and host web applications. With App Engine, developers are able to write and test the application code and then upload it to Google with a simple click of a button or command line script. After the application is uploaded to Google, developers can concentrate on improving and developing new services for their users as Google takes care of the system administration. [35]

App Engine has **built-in automatic scaling**, one does not have to worry about the amount of users or data the application can handle. Google applications are built on GFS(Google File System) and BigTable, which serve data-intensive applications such as Google Earth. By using App Engine, one can count on these highly scalable technologies. [35] There are **no charges** at all in order to publish an application for people to use. All one needs to get started is an account and in the future when the application need to use more resources, billing can be enabled and of course the pay-per-use model makes it possible to pay for only the resources used. 10 applications can be registered per account. “Every Google App Engine application will have enough CPU, bandwidth, and storage to serve around 5 million monthly page views for free.”[35] In order to share the application, App Engine provides a free name on the appspot.com domain. There is also a possibility for users to use their own domain names. [35]
5.2 The Application Environment

Google App Engine supports apps written in a variety of programming languages. With App Engine’s Java runtime environment, one can build an app using standard Java technologies, including the JVM, Java servlets, and the Java programming language or any other language that uses a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features two dedicated Python runtime environments, each of which includes a fast Python interpreter and the Python standard library. Finally, App Engine provides a Go runtime environment that runs natively compiled Go code. These runtime environments are built to ensure that application runs quickly, securely, and without interference from other apps on the system. [35]

5.3 The Data Storage

The App Engine environment provides an array of options for storing data:

1. App Engine Datastore provides a NoSQL schemaless object datastore, with a query engine and atomic transactions.
2. Google Cloud SQL provides a relational SQL database service for the App Engine application, based on the familiar MySQL RDBMS.
3. Google Cloud Storage provides a storage service for objects and files up to terabytes in size, accessible from Python and Java applications. [35]

The application MyNotes uses the datastore. The Java datastore SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces, as well as a low-level datastore API [40]. The datastore holds data objects known as entities. An entity has one or more properties: for instance, a property can be a string, an integer, or a reference to another entity. Each entity is identified by its kind, which categorizes the entity for the purpose of queries, and a key that uniquely identifies it within its kind. The datastore can execute multiple operations in a single transaction.
5.4 App Engine APIs

App Engine allows an app to integrate with Google Accounts for user authentication. Using Google Accounts saves the developer the effort and time he could spend implementing a user account system for the application and it lets the user start using the application faster, because the user does not have to register as most people already have Google accounts. The users API can also tell the application whether the current user is a registered administrator for the application. This makes it easy to implement admin-only areas for the application. [35] There are other numerous APIs available such as OpenID, MapReduce, URLFetch, and XAMPP.

5.5 Development Phases

The App Engine software development kits (SDKs) for Java, Python, and Go emulates all of the App Engine services on one's local computer. Each SDK includes all of the APIs and libraries available on App Engine. The SDK also includes a tool to upload the application to App Engine. When the application has been created, one only has to click on a button to upload it to App engine and it prompts the user for a Google account email address and password. [40]

As soon as a new release of the application that is already running on App Engine is available, one can upload the new release as a new version. The old version will continue to serve users until the administrator switches to the new version. It is possible to test the new version on App Engine while the old version is still running.

The Administration Console is the web-based interface for managing the applications running on App Engine. It can be used to create new applications, configure domain names, change which version of the application is live, examine access and error logs, and browse the application's datastore. [35]

5.6 Choosing an Operating System for Mobile Clients

There is a wide range of mobile phone manufacturers and devices, but they can be classified according to their operating system (OS). Some of the most well-known include Symbian, BlackBerry OS, iOS, Windows Mobile, and Android.
Android based mobile phones use the Android operating system and the applications are generally written in Java. Android-based mobile phones are largely used and there is documentation to help developers build Android applications, therefore Android was selected as the operating system for the mobile clients. Since the cloud platform that will be used in MyNotes app is App Engine, it was already obvious that the OS that will be chosen is the one developed by Google.

5.7 Planning, Building and Testing Process

Eclipse was used as the development environment to create and test MyNotes app. MyNotes was written in Android, so the process started with installing the Android SDK. There is a Google Plugin for Eclipse, which allows Java developers to quickly design, build, and deploy cloud-based applications. The Google Plugin for Eclipse was installed as well, thus getting a complete development environment. The Android SDK provides all the necessary tools to develop Android applications. This includes a compiler, debugger and a device emulator, as well as its own virtual machine to run Android programs.

Android applications consist of different components and can re-use components of other applications. This leads to the concept of a task in Android. An Android application contains activities. An activity represents the visual representation of an Android application. Activities use views to create the user interface and to interact with the user. Views are user interface widgets, for example buttons or text fields and fragments. A fragment encapsulates application code so that it is easier to reuse it and to support different sized devices. The user interface for activities is defined via XML files (layout files). [41]

The application MyNotes will have two activities, the first one to create notes and the second one to display them. Figure 3 illustrates the lifecycle of an Android activity. The entire lifetime of an activity happens between the call to onCreate() and the call to onDestroy(). For example, in case the activity has some action running in the background to download data from the network, it might create it in onCreate() and then stop it in onDestroy().
The visible lifetime of an activity happens between the call to `onStart()` and the call to `onStop()`. During this time, the user can see the activity on-screen and interact with it. `onStop()` is called when a new activity starts and this one is no longer visible. Between the calls; `onResume()` and `onPause()` the activity is in front of all other activities on screen and has user input focus. [40]

Figure 3. The Android activity lifecycle. Reprinted from URL: http://developer.android.com/guide/components/activities.html [40]

As figure3 shows, the activity should start in `onCreate()`, and release all resources in `onDestroy()`. In android, there is no feature of minimizing windows or opening them beside each other as we do while using desktops and laptops. After starting a program in android, there is a possibility to only view one activity at a time. Therefore, as a developer, it is necessary to know the android activity life cycle before developing.
Google App Engine uses the Jetty servlet container to host applications and supports the Java Servlet API. It provides access to databases via Java Data Objects (JDO) and the Java Persistence API (JPA). App Engine uses Google Bigtable as the distributed storage system for persisting application data. Java Persistence API (JPA) is a standard interface for storing objects containing data into a relational database. The standard defines interfaces for annotating Java objects, retrieving objects using queries, and interacting with a database using transactions. The App Engine Java SDK includes an implementation of JPA for the App Engine datastore. To use JPA to access the datastore, App Engine app needs a configuration file named persistence.xml that must be in the app's war/WEB-INF/classes/META-INF/ directory, with configuration that tells JPA to use the App Engine datastore. [35]

6 Results

In the first part of the development process, a complete mobile application that uses local file storage was created. In the second part, cloud-based services were built and added to the existing Android application in order for it to use the cloud as a backend.

6.1 MyNotes Android Application

MyNotes Android application consists of different classes:

- CreateNote.java: It lets the user type and save notes
- DisplayNotes.java: It displays a list of saved notes
- Note.java: is a single note
- NoteDAO.java: is an interface that specifies the way to add and remove notes from storage
- FileDAO.java: is an implementation that adds and removes notes to a local file
- AppEngineDAO.java: contains an implementation of a NoteDAO for saving Note objects to the App Engine

NoteDAO contains the methods; void add(Note note), void remove(Note note) and ArrayList<Note> getAll().AppEngineDAO.java was added to MyNotes android app after creating the cloud-based services.
In order to explore more with Android and incorporate more functionality within the application MyNotes, the Text-To-Speech (TTS) feature was implemented, as well as the ability to delete notes on a long click. The TTS enables the Android device to speak written text.

The AndroidManifest.xml file defines the application’s components and services. It also contains the required permissions for the application. The application MyNotes requires network access, therefore it was specified in the AndroidManifest.xml file. The main.xml defines the layout of the application. The layout is generally the Graphical User Interface (GUI), which the user interacts with. The layout design for Android can be done either dynamically with Java or with XML. MyNotes layout was done with XML; it has a vertical LinearLayout. EditText was used for text input and two Buttons are implemented in a horizontal LinearLayout. Figure 4 shows the layout of the application MyNotes on the device (emulator droidX). As figure 4 shows, MyNotes consists of two android activities; the user can save notes and then display them.

Figure 4. Layout of MyNotes

As figure 4 illustrates, there is only a button to save a note and another one to show a list of saved notes. The blank space is for entering notes to be saved. Removing a note from the list wasn’t implemented as an activity, but it is carried out in DisplayNotes.java where a note is deleted after a long click on that particular note. DisplayNotes.java, CreateNote.java, Note.java, NoteDAO.java, FileDAO.java, AppEngineDAO.java are presented in appendix 8. After MyNotes app was in place, the testing phase followed.
As shown in figure 5, when writing files locally on the device, clicking on the Show List button, displays the new notes. Figure 5 displays the notes written by the user and they can also be heard since the TTS is implemented. A note also consists of the date and time when it was created, as well as the latitude and longitude where the note was written. Latitude and longitude appear as zero because there is no GPS implementation.

Figure 5. Device view

As illustrated in figure 5, the Show List button shows the created notes and the time they were written. The user can also delete notes that are no longer needed by clicking longer on a specific note. When MyNotes app with local storage was completed, the second part, which consisted of creating cloud-based services followed.

6.2 Cloud-based Services

First, an application identifier was created on http://appengine.google.com. A Google account was required before creating an application identifier. The App Engine plugin was used and the cloud-based services were created using Java Servlets and JPA. Three servlets were implemented; storenote that stores notes to cloud datastore, displaynotes that retrieves notes from the cloud datastore to the client and deletenote that deletes a user’s note from the cloud datastore.
When App Engine receives a web request for any application, it invokes the servlet that corresponds to the URL, as described in the application’s web.xml file in the WEB-INF/ directory. The web.xml implementation can be found in appendix 9.web.xml describes the standard web deployment. In web.xml which is the deployment descriptor, the URL pattern for the application was entered: http://mynotesappid.appspot.com/storenote. Figure 6 illustrates the processes involved in the development plan.

Figure 6. Communication process between Web client and database
As figure 6 illustrates, communication between the Cloud datastore and the client is through Java servlets and JPA. After creating the application ID: mynotesappid, the free domain name https://mynotesappid.appspot.com was now available for use. The application ID was written in the file web/WEB-INF/appengine-web.xml.
6.2.1 Storenote Servlet

By creating a web application project StoreNote, which is the cloud backend; the servlet storenoteservlet.java was automatically generated. doGet() method is used to intercept http GET requests. The development design was in different steps:

- insert servlet mapping in web.xml for /storenote
- deploy StoreNote to App Engine
- test https://mynotesappid.appspot.com/storenote from browser
- show datastore in Admin Console on App Engine
- implement code for storage to data store

The file appengine-web.xml is where the application identifier was entered for the application to be deployed to app engine. With a single click in Eclipse, the app is deployed to app engine. Figure 7 shows the deployment process to Google App Engine.

As shown in figure 7, deploying StoreNote web application project to App Engine is done in a simple click and it is even possible to view the deployment process just in case there might be errors. The first deployment attempt was unsuccessful because the application identifier in the file appengine-web.xml was wrong. After correcting the error, the deployment process began.
Figure 8 shows that the application StoreNote is successfully deployed to app engine.

![Figure 8. Successful deployment to Google App Engine](image)

Deploying to app engine, was followed by a test by entering the URL for StoreNoteServlet; http://mynotesappid.appspot.com/storenote. Figure 9 demonstrates that the storenote servlet works because by entering the URL in the browser, the message specified in the StoreNoteServlet.java implementation was outputted. Figure 9 proves that App engine is surely working. It outputs a note implemented in Note.java, which has a class that represents a single text note.

![Figure 9. Browser screenshot of storenote servlet](image)
In figure 9, every note has a latitude and longitude where the note was created, a date when the note was created and the note itself. Note.java implementation can be found in Appendix 1. The development process continued with developing server JPA code for Object-relational mapping (ORM) and storage to data store. ORM is the process of mapping Java objects to database tables. The Java Persistence API (JPA) is one possible approach to ORM. The JPA can be used to store, update, retrieve, and delete data from relational databases to Java objects and vice versa. [43] JPA connection to Google Big Table is defined by persistence metadata. JPA defines the metadata by the use of annotations and an XML configuration: persistence.xml. The implementation can be found in appendix 2. The next part was EMF.java: a class that creates a single instance of a JPA entity factory. EMF.java was acquired from Programming Google App Engine and can be found in appendix 3. [44]

NoteJPA.java was implemented next. Objects that are to be persisted and queried must have annotations on their field; @Entity, @Id. Each object must have a unique key and a @Id annotation. JPA allows to auto generate the primary key in the database through the @GeneratedValue annotation. NoteJPA.java is a class to store and retrieve notes from the datastore. NoteJPA.java can be found in Appendix 4.

The following was NoteDAO.java: It allows saving a Note to a persistent data store as well as retrieving notes as an ArrayList. NoteDAO.java is implemented in appendix 5.

The following written was NoteDAOJPA.java: It enables to store a Note to Google BigTable using JPA API. NoteDAOJPA.java is shown in appendix 6.

The next task was to modify the previously generated StoreNoteServlet doGet and add doPost, to create Note that will be passed to the URL. This will be used as a test. Figure 10 shows passing a message to the URL as a test.
As shown in figure 10, by passing a message to the URL and seeing that the storenote servlet outputs the message, I was able to know that the servlet works.

By entering the URL: mynotesappid.appspot.com/storenote?message=App Engine is the best, the storenote servlet outputs the message. The spaces in the message: App Engine is the best are encoded as %20 when entered in the browser. The same messages can also be seen through the app engine admin console in the datastore viewer. Figure 11 is a view of the admin console demonstrating the different entries made. As figure 11 shows, all the messages are displayed, and can be deleted as well.
As shown in figure 11, the number of entities from the datastore viewer in App Engine Admin console. The message passed to the URL is also included. The next task was NoteUtils.java, which implements a class with methods shared by servlets. NoteUtils.java is shown in appendix 7, it has a method getNote to read a serialized Java object from InputStream and return it. A Java serialized object is a mechanism by which object can be represented as a sequence of bytes that includes the object’s data along with information about the object’s type and the types of data stored in the object. App Engine uses the Java Servlet API to provide the request data to the servlet, and accept the response data. Two more servlets for the application MyNotes were created: DisplayNotesServlet.java and DeleteNoteServlet.java

6.2.2 Displaynotes Servlet

Displaynotes Servlet outputs Notes from the datastore and it was implemented in the following steps:

- create a servlet called DisplayNotesServlet
- implement doGet method in DisplayNotesServlet
- insert servlet mapping in web.xml for /displaynotes
- deploy StoreNote project to App Engine
- test https://mynotesappid.appspot.com/displaynotes from browser

6.2.3 Deletenote Servlet

Deletenote Servlet receives a Java serialized object from the application and deletes it from the datastore. It was implemented in the following steps:

- create a servlet called DeleteNoteServlet
- implement doGet/doPost method in DeleteNoteServlet
- insert servlet mapping in web.xml for /deletenote
- deploy StoreNote project to app engine
- test https://mynotesappid.appspot.com/deletenote from browser

In deployment descriptor web.xml, servlet mappings for /displaynotes and /deletenote were added.
The goal was to be able to use App Engine as the cloud backend, and this was achieved. By writing notes on the android device, they will automatically be saved to app engine datastore and if they are deleted on the device, they will be erased on the datastore as well. It is also possible to delete notes through the admin console. Figure 12 shows the datastore view of notes created locally on the device as well as notes created in the browser.

As shown in figure 12, the Google App Engine administration console gives the developer complete access to the public version of the application. The application gave the expected results and the goal to build an application that uses app engine as backend storage was reached. App Engine is suitable for various types of applications, it is easy to get started and it would not take long for a skilled developer to be efficient on this platform.
7 Discussion

The mobile application MyNotes showed an implementation of a Google app engine datastore as a backend to take some workload off of the mobile device. The answer to the main question if cloud computing could be used in mobile phone applications to execute some functions on the cloud was answered. Cloud computing can be used to improve the computing ability of the mobile device in terms of storage space or computational abilities.

However, MyNotes app only represents a certain aspect of possible tasks that could be offloaded to the cloud. The tests did not include large computations where the capacity of mobile phones would not be enough. The tasks carried out did not require a lot of processing power and could be executed by a mobile phone. For future work, it would be beneficial to test and investigate how a heavy task which consumes a lot of the mobile device’s resources or is not even possible to be performed on mobile devices, would use the cloud services to run on a mobile device. A further topic to consider is to explore building mobile applications on other cloud computing platforms. In this project Google App Engine was used but there are other cloud vendors with different types of services.

8 Conclusion

The goal of the project was to use cloud computing to improve the computing ability of mobile phones, and this was proven to be possible by implementing a mobile phone application that uses Google App Engine as storage. Implementation of cloud computing in mobile applications has several advantages since it combines the advantages of both mobile computing and cloud computing, thus providing ideal services for mobile users. This project provided an understanding of mobile cloud computing and the process of implementing cloud computing in mobile applications.

From the application developed, MyNotes, one is be able to create a new Note, view saved notes and delete them when no longer needed. The project enabled me to get a clear understanding of cloud computing. The process of developing an application in Android also taught me new skills in Android development and I also learned to program the Google app engine.
Using Eclipse as a development environment also gave me a better understanding of the different software development kits needed to achieve different functionalities. It was instructive for me to look into a PaaS solution like Google App Engine and it also gave me the opportunity to work with other technologies such as Android and the Google App Engine datastore. Briefly, I was able to learn in the process and the project reached its goal, developing a mobile application that uses the cloud as a backend. Cloud computing enables developers to build more secure, highly scalable, available, reliable, and high-performance platform-independent applications with a shorter time to the market. The cloud-based mobile application market is expanding at an exponential rate and it is a total market changer and a new "industrial revolution".
References


2 Portio Research Mobile Factbook 2011 [online].


19 Schadler T. Talking to your CFO about cloud computing. Forrester Reasearch Inc.; 2008.


Appendices

Appendix 1 Note.java

```java
package com.android.myprojects;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Note implements Serializable{
    private double latitude;
    private double longitude;
    private Date date;
    private String message;

    public Note(){
        this("");
    }

    public Note(double latitude, double longitude, Date date, String message){
        this.latitude = latitude;
        this.longitude = longitude;
        this.date = date;
        this.message = message;
    }

    public Note(Note note){
        this.latitude=note.latitude;
        this.longitude=note.longitude;
        this.date=(Date)note.date.clone();
        this.message=new String(note.message);
    }

    public Note(String message){
        this.message = message;
        latitude=0.0;
        longitude=0.0;
        date=new Date();
    }

    @Override
    public boolean equals(Object obj){
        if(this==obj){
            return true;
        }
        if(obj==null){
            return false;
        }
        if(getClass()!=obj.getClass()){
            return false;
        }
        return false;
    }
}
```
Note other = (Note) obj;
if (date == null) {
    if (other.date == null) {
        return false;
    } else if (!date.equals(other.date)) {
        return false;
    }
    if (Double.doubleToLongBits(latitude) != Double.doubleToLongBits(other.latitude)) {
        return false;
    }
    if (Double.doubleToLongBits(longitude) != Double.doubleToLongBits(other.longitude)) {
        return false;
    }
    if (message == null) {
        if (other.message == null) {
            return false;
        } else if (!message.equals(other.message)) {
            return false;
        }
    }
    return true;
}

public Date getDate() {
    return date;
}

public double getLatitude() {
    return latitude;
}

public double getLongitude() {
    return longitude;
}

public String getMessage() {
    return message;
}

public void setDate(Date date) {
    this.date = date;
}

public void setLatitude(double latitude) {
    this.latitude = latitude;
}

public void setLongitude(double longitude) {
    this.longitude = longitude;
}

public void setMessage(String message) {
    this.message = message;
}
@Override
public String toString()
{
    String newLine="\n";
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss' z'");
    String s = "Lat: +getLatitude()+newLine;
    s+="Long: +getLongitude()+newLine;
    s+="Date: +dateFormat.format(this.getDate())+newLine;
    s+="Message: +getMessage();
    return s;
}

Appendix 2 persistence.xml

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd" version="1.0">
    <persistence-unit name="transactions-optional">
        <provider>org.datanucleus.api.jpa.PersistenceProviderImpl</provider>
        <properties>
            <property name="datanucleus.NontransactionalRead" value="true"/>
            <property name="datanucleus.NontransactionalWrite" value="true"/>
            <property name="datanucleus.ConnectionURL" value="appengine"/>
        </properties>
    </persistence-unit>
</persistence>

Appendix 3 EMF.java
package com.android.myprojects;

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public final class EMF {
    private static final EntityManagerFactory emfInstance = Persistence
                .createEntityManagerFactory("transactions-optional");

    private EMF()
    {
    }

    public static EntityManagerFactory get()
    {
        return emfInstance;
    }
}

Appendix 4 NoteJPA.java
package com.android.myprojects;
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class NoteJPA implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private double latitude;
    private double longitude;
    private Date date;
    private String message;

    public NoteJPA() {
        id = null;
        latitude = 0.0;
        longitude = 0.0;
        date = new Date();
        message = "Hello, this is a test message!";
    }

    public NoteJPA(double latitude, double longitude, Date aDate, String aMessage) {
        super();
        this.latitude = latitude;
        this.longitude = longitude;
        date = aDate;
        message = aMessage;
    }

    public NoteJPA(Note aNote) {
        id = null;
        setLatitude(aNote.getLatitude());
        setLongitude(aNote.getLongitude());
        setDate(aNote.getDate());
        setMessage(aNote.getMessage());
    }

    public Date getDate() {
        return date;
    }

    public double getLatitude() {
        return latitude;
    }
}
public double getLongitude() {
    return longitude;
}

public String getMessage() {
    return message;
}

public Note getNote() {
    Note result = new Note(message);
    result.setLatitude(latitude);
    result.setLongitude(longitude);
    result.setDate(date);
    return result;
}

public void setDate(Date aDate) {
    date = aDate;
}

public void setLatitude(double latitude) {
    this.latitude = latitude;
}

public void setLongitude(double longitude) {
    this.longitude = longitude;
}

public void setMessage(String aMessage) {
    message = aMessage;
}

@Override
public String toString() {
    String newLine = "\n";
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    String s = "Lat:\" + getLatitude() + newLine;
    s += "Long:\" + getLongitude() + newLine;
    s += "Date:\" + dateFormat.format(this.getDate()) + newLine;
    s += "Message:\" + getMessage();
    return s;
}

Appendix 5 NoteDAO.java

package com.android.myprojects;

import java.io.Serializable;
import java.util.ArrayList;
public interface NoteDAO extends Serializable {
    public void add(Note note);
    public ArrayList<Note> getAll();
    public void remove(Note note);
}

Appendix 6 NoteDAOJPA.java
package com.android.myprojects;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Query;

public class NoteDAOJPA implements NoteDAO {
    @Override
    public void add(Note aNote) {
        EntityManager em = null;
        try {
            em = EMF.get().createEntityManager();
            NoteJPA aNoteJPA = new NoteJPA(aNote);
            em.persist(aNoteJPA);
        } finally {
            em.close();
        }
    }

    @SuppressWarnings("unchecked")
    @Override
    public ArrayList<Note> getAll() {
        EntityManager em = null;
        List<NoteJPA> dbNotes = null;
        ArrayList<Note> notes = new ArrayList<Note>чки();
        String aQuery = "SELECT n FROM NoteJPA n";
        try {
            em = EMF.get().createEntityManager();
            Query query = em.createQuery(aQuery);
            dbNotes = query.getResultList();
            if ((dbNotes != null) && (dbNotes.size() > 0)) {
                for (NoteJPA aNoteJPA : dbNotes) {
                    notes.add(aNoteJPA.getNote());
                }
            }
        } finally {
            em.close();
        }
        return notes;
    }

}
@Override
public void remove(Note note) {
    EntityManager em = null;
    String aQuery = "DELETE FROM NoteJPA n WHERE n.message = :message";
    try {
        em = EMF.get().createEntityManager();
        Query query = em.createQuery(aQuery);
        query.setParameter("message", note.getMessage());
        int x = query.executeUpdate();
    } finally {
        em.close();
    }
}

NoteUtils.java
package com.android.myprojects;
import java.io.InputStream;
import java.io.ObjectInputStream;
public class NoteUtils {

    public NoteUtils() {
        // intentionally left blank
    }

    public static Note getNote(InputStream ais) {
        Note result = null;
        ObjectInputStream ois = null;
        try {
            ois = new ObjectInputStream(ais);
            Object aObject = ois.readObject();
            result = (Note) aObject;
        } catch (ClassNotFoundException e) {
            result = null;
        } catch (Exception e) {
            result = null;
        }
        return result;
    }
}

CreateNote.java
package com.android.myprojects;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
Appendix 8

```java
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class CreateNote extends Activity {
    private NoteDAO noteDAO;
    private EditText et;
    private LocationManager locMgr;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_note);

        noteDAO = new AppEngineDAO();

        et = (EditText) findViewById(R.id.note);

        Button save = (Button) findViewById(R.id.save);
        Button show = (Button) findViewById(R.id.show);

        // attach ButtonListeners to the Buttons
        save.setOnClickListener(saveButtonListener);
        show.setOnClickListener(showButtonListener);

        // request access to the shared location service
        locMgr = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    }

    @Override
    protected void onResume() {
        super.onResume();
        locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locListener);
    }

    @Override
    protected void onPause() {
        super.onPause();
        locMgr.removeUpdates(locListener);
    }

    private LocationListener locListener = new LocationListener() {
        public void onLocationChanged(Location location) {
        }
        public void onProviderDisabled(String provider) {
        }
        public void onProviderEnabled(String provider) {
        }
        public void onStatusChanged(String provider, int status, Bundle extras) {
        }
    }
```
private OnClickListener saveButtonListener = new OnClickListener() {
    public void onClick(View v) {
        if (et.getText().toString().length() > 0) {
            Note note = new Note(et.getText().toString());
            Location loc = locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if (loc != null) {
                note.setLatitude(loc.getLatitude());
                note.setLongitude(loc.getLongitude());
                noteDAO.add(note);
            } else {
                noteDAO.add(note);
            }
            et.setText("");
        }
    }
}

private OnClickListener showButtonListener = new OnClickListener() {
    public void onClick(View v) {
        startActivity(new Intent(CreateNote.this, DisplayNotes.class));
    }
};

DisplayNotes.java

package com.android.myprojects;

import com.android.myprojects.AppEngineDAO;
import com.android.myprojects.NoteDAO;
import android.app.ListActivity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ListView;
import android.widget.ArrayAdapter;

public class DisplayNotes extends ListActivity implements TextToSpeech.OnInitListener{
    private NoteDAO noteDAO;
    private TextToSpeech tts;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //noteDAO = new FileDAO(this);
        noteDAO = new AppEngineDAO();
    }

    private OnClickListener saveButtonListener = new OnClickListener() {
        public void onClick(View v) {
            if (et.getText().toString().length() > 0) {
                Note note = new Note(et.getText().toString());
                Location loc = locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                if (loc != null) {
                    note.setLatitude(loc.getLatitude());
                    note.setLongitude(loc.getLongitude());
                    noteDAO.add(note);
                } else {
                    noteDAO.add(note);
                }
                et.setText("");
            }
        }
    }

    private OnClickListener showButtonListener = new OnClickListener() {
        public void onClick(View v) {
            startActivity(new Intent(CreateNote.this, DisplayNotes.class));
        }
    };
// fills the list with data from NoteDAO
setListAdapter(new ArrayAdapter<>(this,
    android.R.layout.simple_list_item_1,noteDAO.getAll()));

// setup a long-click listener
getListView().setOnItemLongClickListener(longClickListener);

// setup a TTS object
tts=new TextToSpeech(this,this);
}

// what to do when an item is clicked
@Override
public void onListItemClick(ListView parent, View v, int position, long id){
    Note note=noteDAO.getAll().get(position);
    tts.speak(note.getMessage(),TextToSpeech.QUEUE_FLUSH,null);
}

// when app exits, shutdown TTS
@Override
public void onDestroy(){
    super.onDestroy();
    if(tts!=null){
        tts.stop();
        tts.shutdown();
    }
}

// update the adapter
private void updateAdapter(){
    setListAdapter(new ArrayAdapter<>(this,
    android.R.layout.simple_list_item_1,noteDAO.getAll()));
}

private OnItemLongClickListener longClickListener=new OnItemLongClickListener(){
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id){
        Note note=noteDAO.getAll().get(position);
        noteDAO.remove(note);
        updateAdapter();
        return true;
    }
};

public void onInit(int status){}
• NoteDAO.java

```java
package com.android.myprojects;

import java.util.ArrayList;
import java.io.Serializable;

public interface NoteDAO extends Serializable {
    public void add(Note note);
    public void remove(Note note);
    public ArrayList<Note> getAll();
}
```

• FileDAO.java

```java
package com.android.myprojects;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.StreamCorruptedException;
import java.util.ArrayList;
import android.content.Context;

public class FileDAO implements NoteDAO {
    private Context context;
    private final String FILENAME = "notecloud.dao";

    public FileDAO(Context context) {
        this.context = context;
    }

    private ArrayList<Note> createArrayList() {
        // check to see if the file already exists
        File file = new File(context.getFilesDir() + "\" + FILENAME);
        if (!file.exists()) {
            return new ArrayList<Note>();
        }

        // files exists -- read in the stored ArrayList<Note>
        ArrayList<Note> allNotes = null;
        try {
            FileInputStream fis = context.openFileInput(FILENAME);
            ObjectInputStream reader = new ObjectInputStream(fis);
            allNotes = (ArrayList<Note>) reader.readObject();
            reader.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (StreamCorruptedException e) {
            e.printStackTrace();
        }
    }
}
```
Appendix 12

```java
} catch (StreamCorruptedException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}

return allNotes;

private void saveArrayList(ArrayList<Note> allNotes) {
    ObjectOutputStream writer;
    try {
        FileOutputStream fos = context.openFileOutput(FILENAME, Context.MODE_PRIVATE);
        writer = new ObjectOutputStream(fos);
        writer.writeObject(allNotes);
        writer.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void add(Note note) {
    ArrayList<Note> allNotes = createArrayList();
    allNotes.add(note);
    saveArrayList(allNotes);
}

public void remove(Note note) {
    ArrayList<Note> allNotes = createArrayList();
    allNotes.remove(note);
    saveArrayList(allNotes);
}

public ArrayList<Note> getAll() {
    return createArrayList();
}

• AppEngineDAO.java

package com.android.myprojects;

import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.util.ArrayList;
```

class AppEngineDAO implements NoteDAO {
    private final String BASE_URL = "http://mynotesappid.appspot.com/";
    private final String GET = "displaynotes";
    private final String SAVE = "storenote";
    private final String REMOVE = "deletenote";
    private HttpClient client;

    public AppEngineDAO() {
        client = new DefaultHttpClient();
    }

    public void add(Note note) {
        HttpPost postRequest = new HttpPost(BASE_URL + SAVE);
        try {
            postRequest.setEntity(new SerializableEntity(note, true));
            HttpResponse response = client.execute(postRequest);
            } catch (IOException e) {
                e.printStackTrace();
            }
    }

    public void remove(Note note) {
        HttpPost postRequest = new HttpPost(BASE_URL + REMOVE);
        try {
            postRequest.setEntity(new SerializableEntity(note, true));
            HttpResponse response = client.execute(postRequest);
            } catch (IOException e) {
                e.printStackTrace();
            }
    }

    public ArrayList<Note> getAll() {
        HttpGet getRequest = new HttpGet(BASE_URL + GET);
        try {
            HttpResponse response = client.execute(getRequest);
            InputStream isr = response.getEntity().getContent();
            ObjectInputStream reader = new ObjectInputStream(isr);
            return (ArrayList<Note>) reader.readObject();
            } catch (ClientProtocolException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            return null;
    }
}
Appendix 9 web.xml

```xml
    <servlet>
        <servlet-name>StoreNote</servlet-name>
        <servlet-class>com.android.myprojects.StoreNoteServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>StoreNote</servlet-name>
        <url-pattern>/storenote</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DisplayNotes</servlet-name>
        <servlet-class>com.android.myprojects.DisplayNotesServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DisplayNotes</servlet-name>
        <url-pattern>/displaynotes</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DeleteNote</servlet-name>
        <servlet-class>com.android.myprojects.DeleteNoteServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DeleteNote</servlet-name>
        <url-pattern>/deletenote</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>SystemServiceServlet</servlet-name>
        <servlet-class>com.google.api.server.spi.SystemServiceServlet</servlet-class>
        <init-param>
            <param-name>services</param-name>
            <param-value/></param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>SystemServiceServlet</servlet-name>
        <url-pattern>/_ah/spi/**</url-pattern>
    </servlet-mapping>
</web-app>
```