WAP, and WAP 2.0, Mobile Transaction models, File Systems and Mobility Management, Mobile Device Operating Systems – Special Constraints & Requirements, Mobile Payment System – Security Issues

WAP, and WAP 2.0

The Wireless Application Protocol (WAP) is an open, global specification that empowers mobile users with wireless devices to easily access and interact with information and services instantly.

WAP is a global standard and is not controlled by any single company. Ericsson, Nokia, Motorola, and Unwired Planet founded the **WAP Forum** in the summer of 1997 with the initial purpose of defining an industry-wide specification for developing applications over wireless communications networks. The WAP specifications define a set of protocols in application, session, transaction, security, and transport layers, which enable operators, manufacturers, and applications providers to meet the challenges in advanced wireless service differentiation and fast/flexible service creation.

All solutions must be:

interoperable, i.e., allowing terminals and software from different vendors to communicate with networks from different providers

scaleable, i.e., protocols and services should scale with customer needs and number of customers

efficient, i.e., provision of QoS suited to the characteristics of the wireless and mobile networks

reliable, i.e., provision of a consistent and predictable platform for deploying services;and

secure, i.e., preservation of the integrity of user data, protection of devices and servicesfrom security problems.

Why Choose WAP?

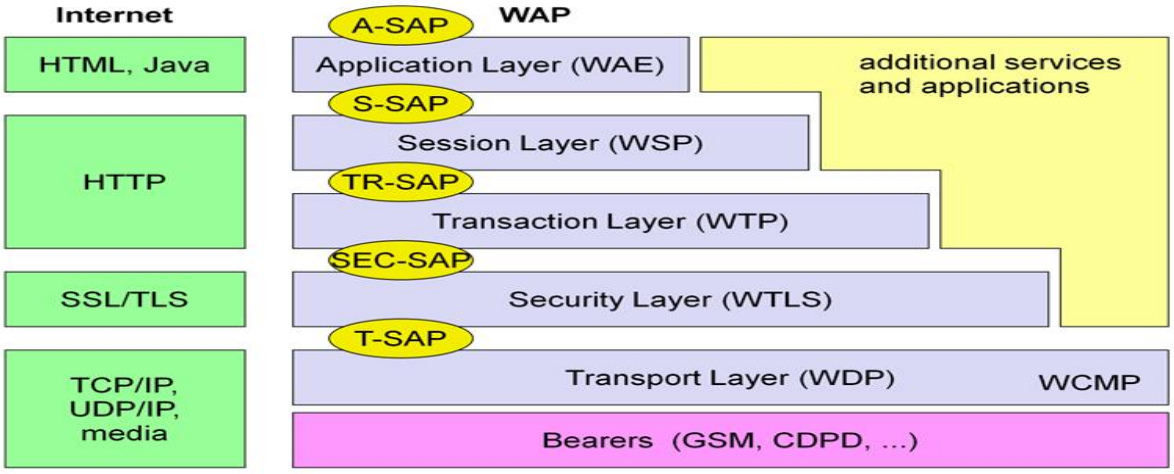
In the past, wireless Internet access has been limited by the capabilities of handheld devices and wireless networks.

1. WAP utilizes Internet standards such as XML, user datagram protocol (UDP), and Internet protocol (IP). Many of the protocols are based on Internet standards such as hypertext transfer protocol (HTTP) and TLS but have been optimized for the unique constraints of the wireless environment: low bandwidth, high latency, and less connection stability.
2. Internet standards such as hypertext markup language (HTML), HTTP, TLS and transmission control protocol (TCP) are inefficient over mobile networks, requiring large amounts of mainly text-based data to be sent. Standard HTML content cannot be effectively displayed on the small-size screens of pocket-sized mobile phones and pagers.

3. WAP utilizes binary transmission for greater compression of data and is optimized for long latency and low bandwidth. WAP sessions cope with intermittent coverage and can operate over a wide variety of wireless transports.
4. WML and wireless markup language script (WML Script) are used to produce WAP content.They make optimum use of small displays, and navigation may be performed with one hand.WAP content is scalable from a two-line text display on a basic device to a full graphic screen on the latest smart phones and communicators.
5. The lightweight WAP protocol stack is designed to minimize the required bandwidth and maximize the number of wireless network types that can deliver WAP content. Multiple networks will be targeted, with the additional aim of targeting multiple networks. These include global system for mobile communications (GSM) 900, 1,800, and 1,900 MHz; interim standard (IS)–136; digital European cordless communication (DECT); time-division multiple access (TDMA), personal communications service (PCS), FLEX, and code division multiple access (CDMA). All network technologies and bearers will also be supported, including short message service (SMS), USSD, circuit-switched cellular data (CSD), cellular digital packet data (CDPD), and general packet radio service (GPRS).
6. As WAP is based on a scalable layered architecture, each layer can develop independently of the others. This makes it possible to introduce new bearers or to use new transport protocols without major changes in the other layers.
7. WAP will provide multiple applications, for business and customer markets such as banking,corporate database access, and a messaging interface.

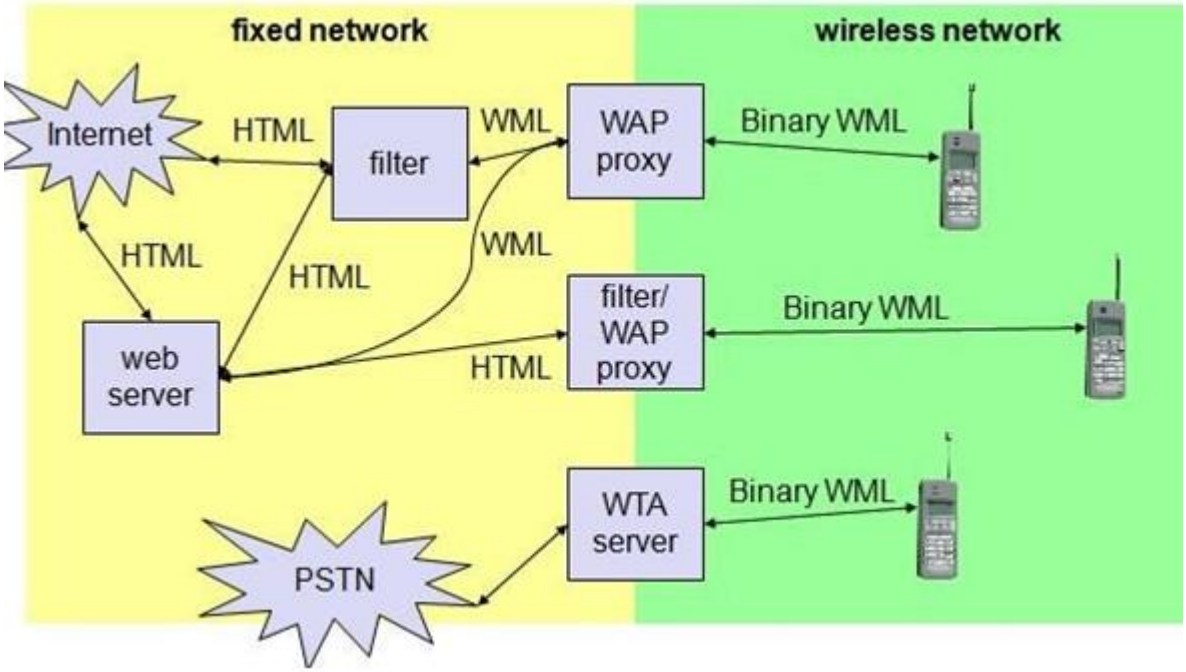
WAP Architecture

The following figure gives an overview of the WAP architecture, its protocols and components, and compares this architecture with the typical internet architecture when using the World Wide Web. The basis for transmission of data is formed by different **bearer services**. WAP does not specify bearer services, but uses existing data services and will integrate further services. Examples are message services, such as short message service (SMS) of GSM, circuit- switched data, such as high-speed circuit switched data (HSCSD) in GSM, or packet switched data, such as general packet radio service (GPRS) in GSM. Many other bearers are supported, such as CDPD, IS-136, PHS.



Working of WAP

WAP does not always force all applications to use the whole protocol architecture. Applications can use only a part of the architecture. For example, if an application does not require security but needs the reliable transport of data, it can **directly** use a service of the transaction layer. Simple applications can directly use WDP.



Different scenarios are possible for the integration of WAP components into existing wireless and fixed networks. On the left side, different fixed networks, such as the traditional internet and the public switched telephone network (PSTN), are shown. One cannot change protocols and services of these existing networks so several new elements will be implemented between these networks and the WAP- enabled wireless, mobile devices in a wireless network on the right-hand side.

The WAP Forum recently released the next revision (Version 2.0) of specifications for the Wireless Application Protocol (WAP). As an evolution of the open wireless standards for delivering applications to mobile devices such as cellular phones and personal digital assistants, WAP 2.0 sounds promising.

WAP 2.0 now supports XHTML (Extensible Hypertext Markup Language) Basic by providing the Wireless Markup Language (WML) as a basic profile WML2. The addition of support for XHTML (the XML version of the popular HTML tagging language) is a good step in the unification of the various presentation formats for Web and wireless delivery. The WAP Forum has also ensured backward compatibility with WML1 (part of the WAP 1.2 specification).

This is particularly important because part of the WAP stack -- the Wireless Application Environment --lives on the device, and it can take quite some time for the new stack to be available on new handsets.

A significant aspect of the WAP 2.0 specification is its support for the popular Web protocols such as TCP/IP, TLS and HTTP. In the previous versions of the WAP specification, a new set of protocols, collectively known as WAP 1 Stack, were created to make use of low-bandwidth mobile networks. This stack included the Wireless Session Protocol, Wireless Transaction Protocol, Wireless Transport Layer Security and Wireless Datagram Protocol. With the advent of the next-generation 2.5G and 3G high-speed wireless networks, it has become increasingly important to support the same set of protocols that are available on the Web. As another measure to ensure backward compatibility, manufacturers of devices and microbrowsers can choose to implement a dual stack to support both sets of protocols.

Other features of WAP 2.0 include enhancement of WAP Push functionality, User Agent Profile support for describing the device capabilities, an External Functionality Interface to support external plug-in like functionality in microbrowsers, support for SyncML for data synchronization, support for persistence capability in a device, small pictures (called pictogram) and multimedia messaging service to deliver multimedia content to the device.

A number of key vendors have announced support for the WAP 2.0 standard. However, widespread usage for applications based on WAP 2.0 will surface only when development tools, microbrowsers, gateways and handsets are available with WAP 2.0 support. Unlike desktop Web browsers, upgrading wireless microbrowsers in a handset it isn't a straightforward task. Most consumers will have to wait until new WAP 2.0-compatible handsets are available.

MOBILE TRANSACTION MODELS

The disconnection of mobile stations for possibly long periods of time and bandwidth limitations requires a serious reevaluation of transaction model and transaction processing techniques. There have been many proposals to model mobile transactions with different notions of a mobile transaction. Most of these approaches view a mobile transaction as consisting of sub transactions which have some flexibility in consistency and commit processing. The management of these transactions may be static at the mobile unit or the database server, or may move from base station to base station as the mobile unit moves. Network disconnection may not be treated as failure, and if the data and methods needed to complete a task are already present on the mobile device, processing may continue even though disconnection has occurred. Because the traditional techniques for providing serializability (e.g., transaction monitors, scheduler, locks) do not function properly in a disconnected environment, new mechanisms are to be developed for the management of mobile transaction processing Applications of mobile computing may involve many different tasks, which can include long-lived transactions as well as some data-processing tasks as remote order entry . Since users need to be able to work effectively in disconnected state, mobile devices will require some degree of transaction management. So, concurrency control schemes for mobile distributed databases should support the autonomous operation of mobile devices during disconnections. These schemes should also consider the message traffic with the realization of bandwidth limitations. Another issue in these schemes would be to consider the new locality or place after the movement of the mobile device. These challenging issues have been studied by many researchers but only some of the work is included below. In many of these models,

relaxing some of the ACID properties and non-blocking execution in the disconnected mobile unit, and caching of data before the request, adaptation of commit protocols and recovery issues are examined. Each used its basic requirements for the transaction models. However, the first of the following transaction models is a new model especially defined for the mobile environment based on the traditional transaction models.

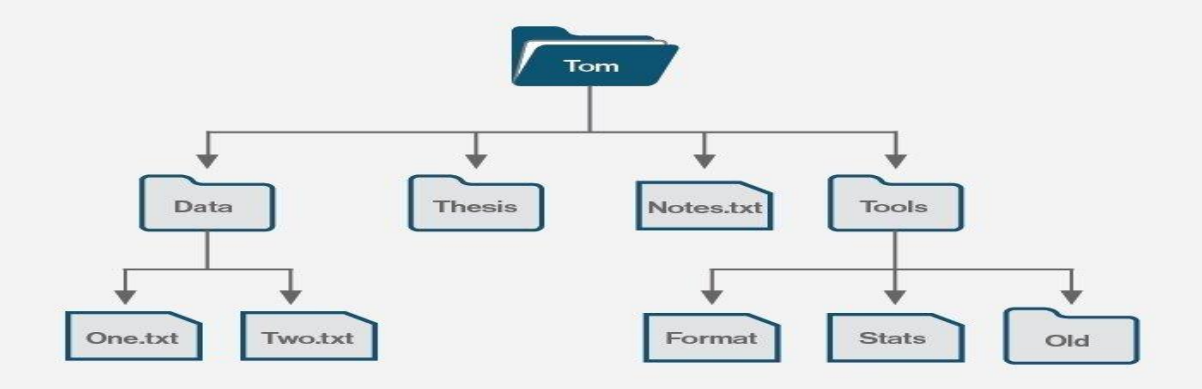
File Systems

In a computer, a file system -- sometimes written filesystem -- is the way in which files are named and where they are placed logically for storage and retrieval. Without a file system, stored information wouldn't be isolated into individual files and would be difficult to identify and retrieve. As data capacities increase, the organization and accessibility of individual files are becoming even more important in data storage. Digital file systems and files are named for and modeled after paper-based filing systems using the same logic-based method of storing and retrieving documents.

File systems can differ between operating systems (OS), such as Microsoft Windows, macOS and Linux-based systems. Some file systems are designed for specific applications. Major types of file systems include distributed file systems, disk-based file systems and special purpose file systems.

How file systems work

A file system stores and organizes data and can be thought of as a type of index for all the data contained in a storage device. These devices can include hard drives, optical drives and flash drives.File systems specify conventions for naming files, including the maximum number of characters in a name, which characters can be used and, in some systems, how long the file name suffix can be. In many file systems, file names are not case sensitive.Along with the file itself, file systems contain information such as the size of the file, as well as its attributes, location and hierarchy in the directory in the metadata. Metadata can also identify free blocks of available storage on the drive and how much space is available.



A file system also includes a format to specify the path to a file through the structure of directories. A file is placed in a directory -- or a folder in Windows OS -- or subdirectory at the desired place in the tree structure. PC and mobile OSes have file systems in which files are placed somewhere in a hierarchical tree structure.Before files and directories are created on the storage medium, partitions should be put into place. A partition is a region of the hard disk or other storage that the OS manages separately. One file system is contained in the primary partition, and some OSes allow for multiple partitions on one disk. In this situation, if one file system gets corrupted, the data in a different partition will be safe.

File systems and the role of metadata

File systems use metadata to store and retrieve files. Examples of metadata tags include:

- Date created
- Date modified
- Last date of access
- Last backup
- User ID of the file creator
- Access permissions
- File size

Metadata is stored separately from the contents of the file, with many file systems storing the file names in separate directory entries. Some metadata may be kept in the directory, whereas other metadata may be kept in a structure called an inode.

In Unix-like operating systems, an inode can store metadata unrelated to the content of the file itself. The inode indexes information by number, which can be used to access the location of the file and then the file itself.

An example of a file system that capitalizes on metadata is OS X, the OS used by Apple. It allows for a number of optimization features, including file names that can stretch to 255 characters.

File system access

File systems can also restrict read and write access to a particular group of users. Passwords are the easiest way to do this. Along with controlling who can modify or read files, restricting access can ensure that data modification is controlled and limited.

File permissions such as access or capability control lists can also be used to moderate file system access. These types of mechanisms are useful to prevent access by regular users, but not as effective against outside intruders.

Encrypting files can also prevent user access, but it is focused more on protecting systems from outside attacks. An encryption key can be applied to unencrypted text to encrypt it, or the key can be used to decrypt encrypted text. Only users with the key can access the file. With encryption, the file system does not need to know the encryption key to manage the data effectively.

Types of file systems

There are a number of types of file systems, all with different logical structures and properties, such as speed and size. The type of file system can differ by OS and the needs of that OS. The three most common PC operating systems are Microsoft Windows, Mac OS X and Linux. Mobile OSes include Apple iOS and Google Android.

Major file systems include the following:

File allocation table (FAT) is supported by the Microsoft Windows OS. FAT is considered simple and reliable, and it is modeled after legacy file systems. FAT was designed in 1977 for floppy disks, but was later adapted for hard disks. While efficient and compatible with most current OSes, FAT cannot match the performance and scalability of more modern file systems.

Global file system (GFS) is a file system for the Linux OS, and it is a shared disk file system. GFS offers direct access to shared block storage and can be used as a local file system.

GFS2 is an updated version with features not included in the original GFS, such as an updated metadata system. Under the terms of the GNU General Public License, both the GFS and GFS2 file systems are available as free software.

Hierarchical file system (HFS) was developed for use with Mac operating systems. HFS can also be referred to as Mac OS Standard, and it was succeeded by Mac OS Extended. Originally introduced in 1985 for floppy and hard disks, HFS replaced the original Macintosh file system. It can also be used on CD-ROMs.

The NT file system -- also known as the **New Technology File System (NTFS)** -- is the default file system for Windows products from Windows NT 3.1 OS onward. Improvements from the previous FAT file system include better metadata support, performance and use of disk space. NTFS is also supported in the Linux OS through a free, open-source NTFS driver. Mac OSes have read-only support for NTFS.

Universal Disk Format (UDF**)** is a vendor-neutral file system used on optical media and DVDs. UDF replaces the ISO 9660 file system and is the official file system for DVD video and audio as chosen by the DVD Forum.

Mobile Device Operating Systems – Special Constraints & Requirements

Special Constraints & Requirements

1. **Screen size, sensors and interactions**

Screen sizes are much smaller when designing mobile software. You have a limited canvas, and your design should be simple with enough space for users to touch and interact with different elements.

Users may use your software with one hand or two hands, or use any of the supported gestures to interact with your software.

Depending on your content, you could choose a spatial format (map view), a list format, a block design or other ways to display your content.

If you are building for iOS or Windows Phone, there are fixed screen sizes and resolutions to plan for. If you're building for Android, you have more variations to keep in mind.

You also have a variety of sensors and enablers that can help you design interactions. While many of them are great enablers for design, they also come with their constraints (eg: GPS when used indoors with a spotty data connection may not return a location. How will your software handle this?)

2. Storage and cache sizes

Depending on what you're building, you might have options to download and store/cache content for offline usage. This could help reduce the data transfer for online transactions, making the product feel more responsive.

Think aboutwhether the storage is available on a memory card (removable storage) or internal memory.

With a memory card, you have to tackle states where the user might delete/modify content on the memory card, pull out the memory card while interacting with the software or for memory card corruption.

With internal software, you may have limited allotment for storage, and the possibility of cached data getting overwritten.

3. Latencies

It's a good idea to time operations and load times. Users on a mobile expect an instant response for something they do.

A lot of this depends on your product architecture. You may be able to do some architectural jugglery to deliver a phased experience. Think of how images load in your browser on a slow connection – a lighter pixelated version loads first, and then the sharper image loads. The intermediate stage keeps the user occupied till the final image loads.

Even if you can't eliminate all latencies, you could plan to tackle the intermediate wait stages with engaging messages or cached content.

First time use is a special case. When the app is used for the very first time, there are usually a lot of background setup activities that need to be done.

One way of mitigating the wait during this time is to have a front-end tutorial for users to engage with, while your app performs setup or bookkeeping in the background. On subsequent usage, you could use data/states cached to deliver a faster experience.

4. Network issues

Any mobile product has to contend with network latencies and failure points. Plan for these earlier on, or you may find you're losing users because they keep staring at a 'Try again later' or 'Connecting…' message.

Try to avoid blocking spinners – one that does not let the user do anything while he waits. See if your platform and design permit loading partial elements and cached data to keep him occupied when he waits.

Depending on the platform you develop for, you may also have cases where the user can pull out the SIM card during an operation and put it back in, or walk in and out of Wi-Fi zones. This affects not just the software on the phone, but also the backend that supports it.

Finally: try identifying failure points and use that as avenues to show your product's personality. Twitter's fail-whale is a great example of how a failure point can become an icon.

5. Data use requirements

In addition to just data connection, the SIM card also provides information like MCC (Mobile Country Code) and MNC (Mobile Network Code) that you may have used to identify which country the user is and what network he is connected to. You can get info on whether he is on a roaming network and home network, and adjust the amount of data required accordingly.

For example, you may choose to download and cache a lot more data when the user is connected to Wi-Fi or a home network, but take a download-as-required approach when he is on roaming.

6. Fonts, language and tone of voice

The fonts and language you use in the product makes a big difference to the user experience.

Language and tone of voice reflect your product's personality.

The tone can be informal (Howdy!), formal (Welcome), friendly (let's get started), impersonal (press this button to continue) or any other variation that suits your product. It's important to realize what tone your customers are comfortable with, and stick to one tone through the product.

Point to note: if you're building a global product, be careful about the tone of voice you choose. What comes across as informal in one culture may seem blasé in another.

Languages may bring their own issues – getting translations, checking that terms do not get curtailed, etc. You may also have to plan for right-to-left languages like Arabic.

7. Corner cases for product usage

There are many use cases that you may not be able to test during development.

For example, it might be difficult for you to test 'international roaming' use cases unless you procure an international SIM, or send an Indian SIM to a tester/friend sitting abroad.

For many of these, it is usually possible to visualize multiple user journeys and scenarios when you're designing the product, and think of which cases you want to implement.

You may decide that there are cases that are too extreme to develop for, which is fine. If you have identified many use cases like this, you could at least plan for these with engaging error/fail messages, or take them as opportunities to get feedback (eg: this product was not designed to work in international roaming scenarios. Send us a quick email if you think we should be focusing on this – with an option for the user to mail quick feedback from the mobile client.

Mobile Payment System – Security Issues

The development of smartphones has gone and replaced a few things we grew up with: the watch, the alarm clock, the tape recorder, music players, and it seems that very soon, we can add cash and wallets to that list. It‘s hardly a surprise. Payment methods have been morphing through various channels: from cash to cheques, to credit cards and debit cards, and now to online banking and mobile commerce.

Close to 10 million mobile subscribers in Japan are already paying for purchases with their smartphones by the end of 2010, and reports are saying that the more than $200 billion dollar mobile payment industry willl be worth a trillion by 2015.

There are 6 billion mobile phone subscriptions in the world, and more than a billion smartphones already in the market. Perhaps it‘s just a matter of time before we embrace the idea of losing that wallet and opting for a digital one to buy flight tickets, lunch, coffee or even to pay the rent.

Digital Wallets

The verdict is still out on what to call these cashless wallets: digital wallet, electronic wallet, e-wallet, virtual wallet etc but they all work the same way. By downloading an app onto your phone, you can link the service or app account to your bank account or payment card. With that done, you can start paying for your wares with your digital wallet.

Paying is a Breeze

If your digital wallet is an NFC enabled Android phone, you can tap your smartphone at the card terminal at the checkout counter, like you would your debit card. But let‘s face it, not all Android phones carry NFC technology and it‘s hardly a strong reason for you to consider when it comes to picking your next smartphone. But fret not, other e-wallets, like Square Wallet, let you pay just by saying your name to the cashier.

Systems like ERPLY allow you to check in at a store, and let the cashier identify you by facial recognition; your purchases are then auto-deducted from your PayPal account.

Restaurants and pubs would love platforms like Tabbedout, which lets their diners check in when they arrive, and pay for their meal anytime without needing to wait for the bill or to bring their wallets along. All of this is made possible with smartphones and the right apps.

Digital Wallets not only carry payment details to allow their owners to make purchases, they also help them to better manage their loyalty cards. If your really want to go full digital (wallet) then it only makes sense that you need not carry around your loyalty cards either.

To cater for this, there are also apps that let users scan the information on the barcodes of their loyalty cards, then store them up in the phone. At the checkout counter, they can let the cashier scan the barcode displayed on their mobile screen to ensure that they don‘t miss out on any rewards.

2. **Loyalty Apps and Programs**

But then other apps take it up a notch and become the reward platform itself. Loyalty platforms like LevelUp, Perka and rewardjunkie! give business owners the flexibility to customize reward programs for their loyal, paying customers, and to engage new customers for their booming business.

For the rest of us, this means that we don‘t have to carry around stacks of brand-specific loyalty cards that are used probably once every couple of months. Everything is in our smartphone, including new offers, discounts and deals offered by participating merchants.

3. **Alternative Payment Methods**

If however you are cautious with your spending and prefer to not put all your chicken eggs in the same basket (i.e. what if you lose your smartphone?), then there are other online payment methods to use.

4. **Carrier or Mobile Billing**

The idea is to charge all your online purchases to your phone bill and clear that at the end of the month. The good thing with this method is that you need not even own a smartphone to start making online purchases. Having a mobile phone is enough as you can pay via sms. There are confirmation codes or authorization pins or text to punch in they are intended for security purposes.

Is it Secure?

Ultimately, the security of these mobile payment systems is always at the back of our heads. What happens if I transfer all my payment card details into the smartphone and the unthinkable happens: someone else gets hold of my lost or stolen smartphone?. Well, it‘s a good thing that most of these accounts, as well as your smartphone, can be remotely deactivated or wiped out. It is a good idea to have a passcode lock, at least to give your phone an extra layer of protection. Also, before you start linking your sensitive data to any mobile payment platform, do take a look at customer reviews or coverage of the platform from reliable sources first.

Resources for accepting mobile payment

To wrap up, here is a small list of resources developers can adapt to their online business to start accepting mobile payments from their online customers.

Card io

Tired of having to punch in line after line of credit card details? You can skip through all that with Card.io by taking a photo of your credit card, then punching in the CVV code manually. This help reduce fraud and developers can easily join the program by grabbing the SDK for card.io at the site.

Jumio

Here is another app that lets you take photos of your credit card as a payment method via Netswipe. It also has a similar online ID verification tool calledNetverify, which lets your customer‘s computer work in your favor as an ID scanning tool.

BancBox

BancBox is an all-in, one-stop solution for businesses that cater to the online marketplace. With the payment portal in place, the business owner can receive credit card payments, wire transfers and checks, among others. It also has a relatively low fee of 0.5% + 30 cents per transaction for its services.

Stripe

Stripe helps developers take care of credit card payments online with a simple JS script. It lets you build your own payment forms, and avoid PCI requirements. Embedding the codes in the site lets Stripe to handle all your online payment needs at 2.9% + 30 cents per successful charge.

Zooz

ZooZ gives developers 3 lines of code, which they can integrate into their mobile applications. There is also a sandbox environment to let developers test out transactions at no charge. Prices are locked in at 2.8% + 19 cents per transaction.