

CIS 5050

Software Systems

Linh Thi Xuan Phan

Department of Computer and Information Science
University of Pennsylvania

Lecture 12: Bigtable + Project Overview

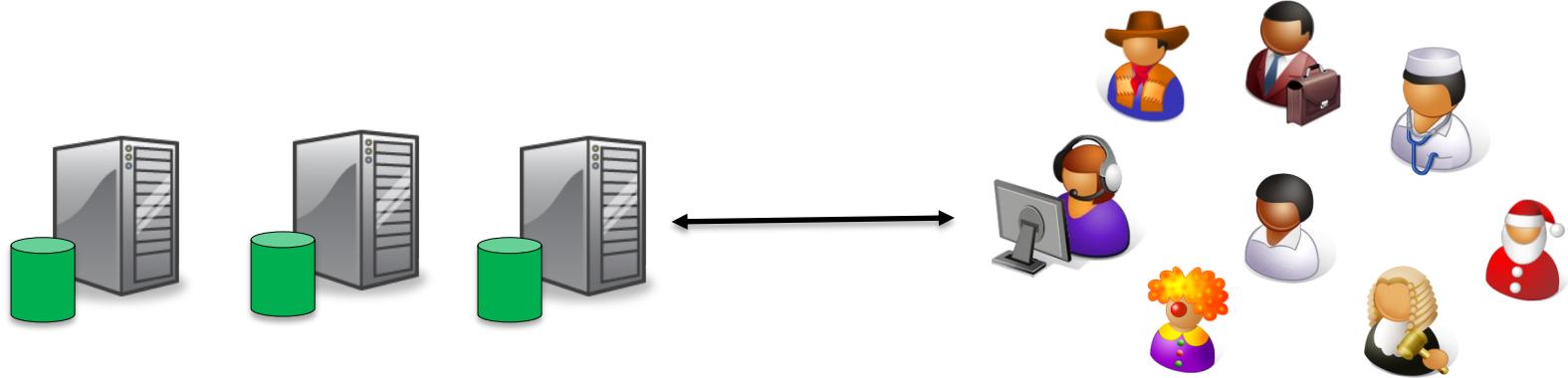
March 21, 2023

Plan for today

- Distributed storage systems
- Bigtable
- A few words about the project



Why distributed storage systems?



- Goal: Make a storage system available to remote users
 - Example: Maintain shared data for a group
 - There could be many users
 - ... or the system itself could be spread across many nodes
- Key goal: Transparency
 - Users should not have to know (or care) where the data is located

How hard can this be?

- This sounds like a relatively simple problem
- And yet there are many many different solutions!
 - Examples: NFS, Coda, Andrew, GFS, Colossus, xFS, Swift, Frangipani, Intermezzo, GPFS, RGFS, Sprite, SMB, Locus, ...
- Why so many?
 - Lots of different design decisions!
 - Examples:
 - How much do we care about scalability? (Data center? Single node?)
 - What kind of consistency do we want?
 - What is the expected workload like?
 - What kind of network connectivity is available?
 - Does fault tolerance matter? If so, how strong do we want it to be?
 - What kind of security guarantees do we need?

Distributed storage: Examples

- Two 'classics'
 - Sun's Network File System (NFS)
 - Widely used! This is the file system you're using if you log into eniac.
 - CMU's Coda file system
 - A research system with interesting features
- Some more scalable examples
 - Google File System (GFS)
 - A file system for data centers; used, e.g., in conjunction with MapReduce/Hadoop
 - Cooperative File System (CFS)
 - A highly scalable file system that is based on a distributed hashtable
 - **Bigtable**
 - A flexible general-purpose storage system for structured data

Plan for today

- Distributed storage systems
- Bigtable
- A few words about the project



Why Bigtable?

- An interesting example of scalable data storage systems with replication
 - Petabytes of data across thousands of machines
- Used extensively in many projects at Google
 - Web indexing, Google Earth, Google Finance, Google Analytics, ...
- Designed for flexibility and performance
 - Varying data sizes (URLs, webpages, satellite images)
 - Often smaller pieces (e.g., compared to GFS); random accesses
 - Structured data (e.g., organized by user/by site/...)
 - Diverse latency requirements (bulk processing, real-time demands)
- Inspiration for the storage system in our project!

Bigtable

- **Bigtable**: A storage system for structured data
 - Designed by Google; widely used by over sixty different projects and products
- Common abstraction: **Key-value pairs**
 - Very general: tuple (k, v) ; k and v can typically be arbitrary bitstrings
 - Example: Key could be an email ID, and value the text of that email
 - Widely used: S3, memcached, MapReduce, DHTs, ...
 - Common interface of **key-value stores**: **PUT(k, v)**, **GET(k) $\rightarrow v$**
- This is the abstraction we will use for our project

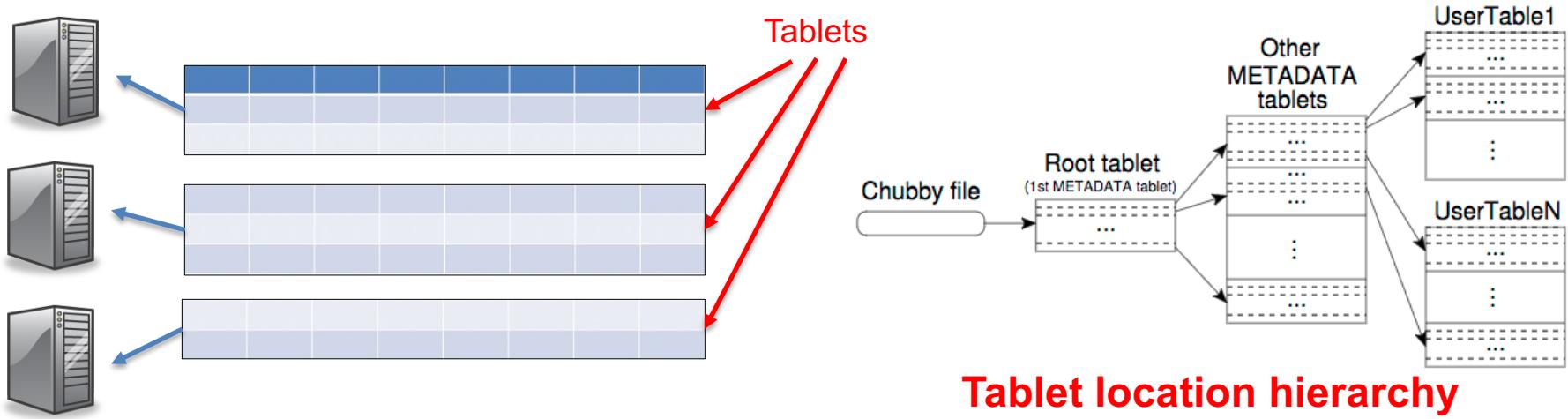
Bigtable

Row key	A	B	C	M	Q
...					
bar	1	2	3		
foo	7	9		8	12
...					

- Provides the abstraction of a giant table
 - Special column ("row key") contains a unique value for each row
 - Each row key is an arbitrary string (up to 64KB, but typical usage 10-100 bytes)
 - Each row is basically a little key-value store of its own – the key identifies a "column" in the row identified by the row key
 - No explicit schema: Different rows can have different columns
 - For instance, row "bar" can contain (A,1), (B,2), (C,3), and row "foo" can contain (A,7), (B,9), (M,8), (Q,12)
 - Each read (write) of data under a row key is atomic. (Why?)

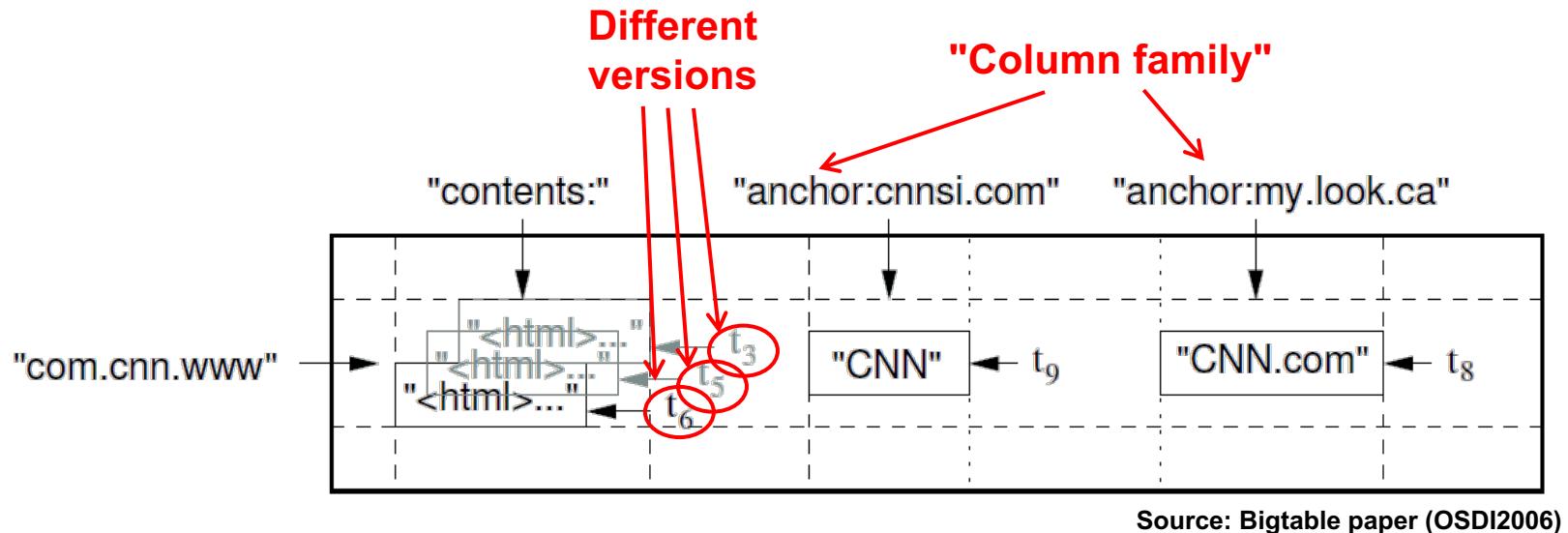
Tablet servers

Tablets in Bigtable



- The table is partitioned into row ranges ("tablets")
 - Tablets are bite-size pieces of data (~100-200MB by default)
- Tablets are distributed across many **tablet servers**
 - Each tablet server has many different tablets at any given time
 - Tablet state is stored in GFS for persistence
- Bigtable maintains an **index** by row key
 - Can be used to find rows by row key

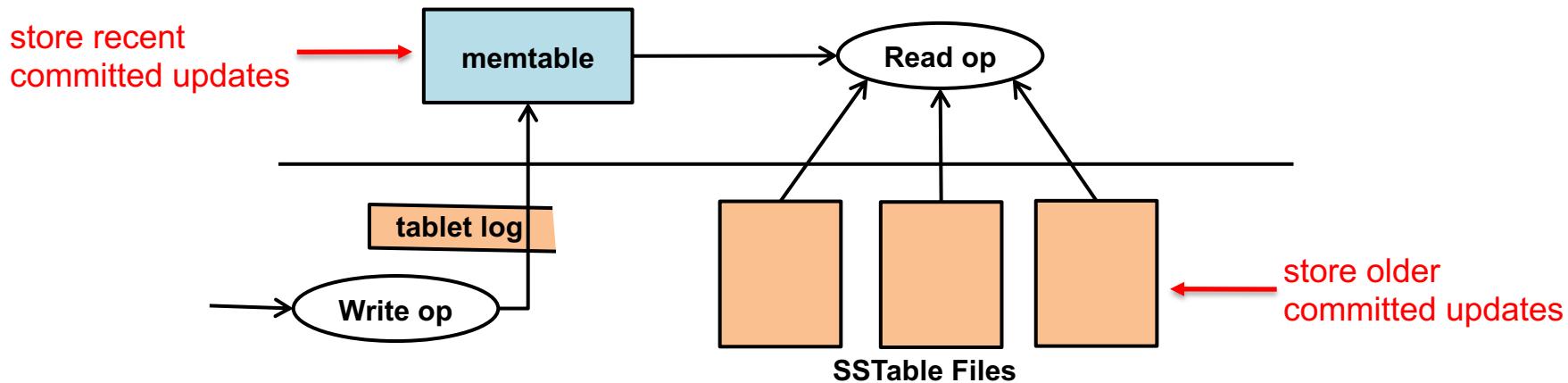
Versioning and column families



Source: Bigtable paper (OSDI2006)

- Data in Bigtable is **versioned**
 - Each cell can 'remember' values it has held previously (by timestamp)
 - Example: Row U, column "contents:" contains crawled web page at URL U
 - Remembering earlier crawls is useful, e.g., to see how a page changes over time
- Columns are grouped into **column families**
 - Example: "anchor:XYZ" columns could contain references from other pages (URL XYZ) to the page in the current row
 - Basic unit of access control

Bigtable implementation



- A **single-master** system, similar to GFS (later lecture)
 - The master assigns tablets to tablet servers, load balancing, etc.
- State is kept in a **memtable** and various **SSTables**
 - Writes are appended to a commit log and then kept in the memtable
 - After a crash, server can restore the state of its memtable by replaying the log
 - The memtable and SSTables are lexicographical sorted data structures
 - When the memtable fills up, it is 'frozen' (made read-only) and written to GFS as a SSTable ("minor compaction")
 - Reads are handled from the memtable and the SSTables
 - A background process occasionally merges SSTables and gets rid of old/deleted data ("major compaction")

Some services that use Bigtable

Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
<i>Crawl</i>	800	11%	1000	16	8	0%	No
<i>Crawl</i>	50	33%	200	2	2	0%	No
<i>Google Analytics</i>	20	29%	10	1	1	0%	Yes
<i>Google Analytics</i>	200	14%	80	1	1	0%	Yes
<i>Google Base</i>	2	31%	10	29	3	15%	Yes
<i>Google Earth</i>	0.5	64%	8	7	2	33%	Yes
<i>Google Earth</i>	70	–	9	8	3	0%	No
<i>Orkut</i>	9	–	0.9	8	5	1%	Yes
<i>Personalized Search</i>	4	47%	6	93	11	5%	Yes

Source: Bigtable paper (OSDI 2006)

- In 2006, there were 388 non-test Bigtable clusters at Google
 - Combined total: 24,500 tablet servers
- Example: Google Analytics
 - Raw click table (~200TB): 1 row for each end-user session
 - Summary table (~20TB): Predefined summaries per website

How does this relate to the class?

- Bigtable uses many of the technologies we will look at later in this course:
 - Lock service is made fault-tolerant with **Paxos**
 - Clients can run per-row **transactions**
 - Data is persisted in a scalable file system, **GFS**
 - Bigtable can be used as source or target for **MapReduce** jobs
- More details are in the paper:
 - F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, R. Gruber: "Bigtable: A Distributed Storage System for Structured Data", OSDI 2006
 - <http://research.google.com/archive/bigtable-osdi06.pdf>
- Inspiration for Apache HBase (open-source)

How does this relate to the project?

- The "key-value store" is modeled after Bigtable
- However, you don't need to implement all features!
 - You can leave out versioning, access control, and the index
 - You can store data directly on the storage node (instead of in GFS)
 - You don't have to use the SSTable mechanism (unless you want to)
- A basic implementation could work like this:
 - Single master that knows which tablets are on which node
 - Several storage nodes that keep their tablets in memory
 - When a PUT happens, the node appends the change to a log file on disk, so it is persistent if the node should crash
 - If a crash does happen, the node can 'replay' the changes in the log file
 - Once in a while, each node writes snapshots of its tablets to a file on disk and then empties its change log

Plan for today

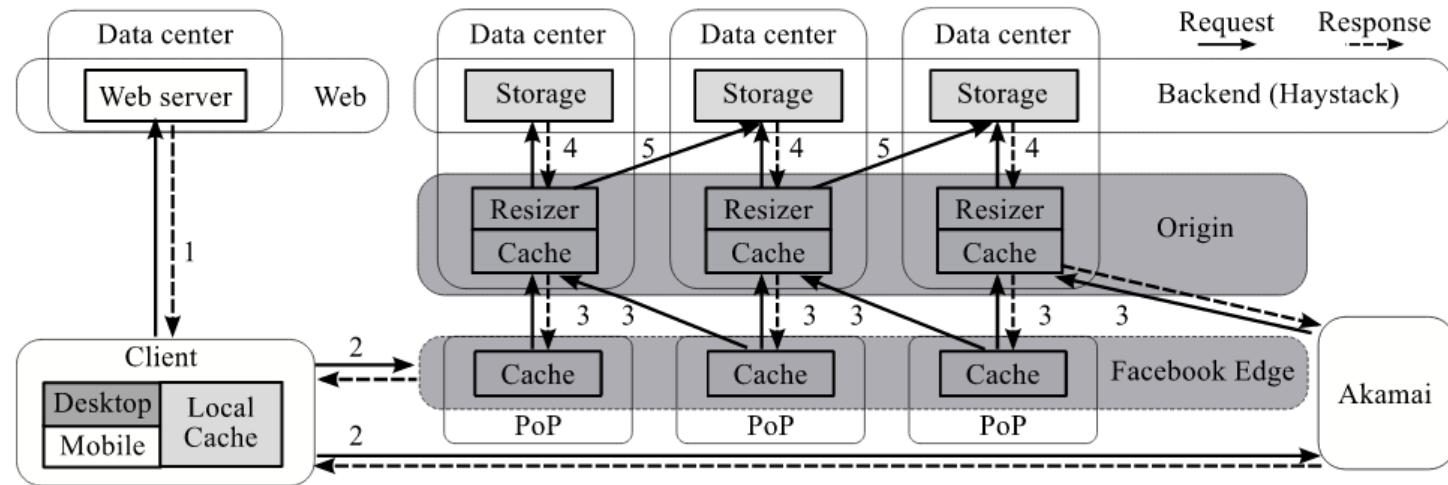
- Distributed storage systems
- Bigtable
- A few words about the project



Why a Final Project?

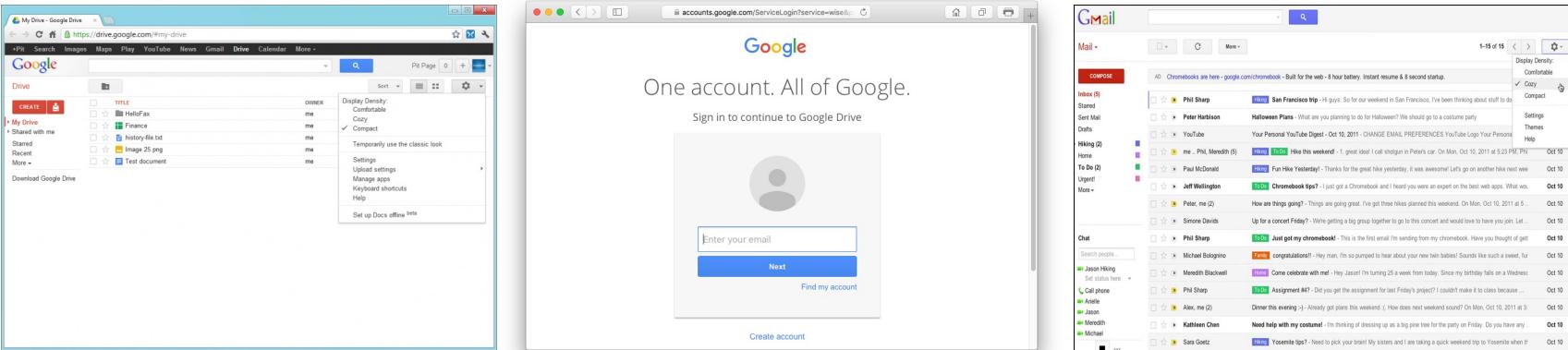
- So far, homeworks have been single components:
 - HW1: Sorting program
 - Opportunity to get experience with processes, threads, and IPC
 - Learn about concurrency, synchronization, coordination
 - HW2: SMTP and POP3
 - Get experience with client/server (very common paradigm!), socket programming
 - Learn to work with, and comply with, simple protocol specifications (RFCs)
 - Understand how simple real-world systems work
 - Why not IMAP? Because it is far more complicated than POP3!
 - HW3: Chat server
 - Get experience with distributed coordination, group communication
 - Work with different consistency models; implement standard ordering protocols
- But the class is about software systems!
 - Don't these have multiple interacting components?

Real-world systems



- Example: Facebook's photo serving stack [SOSP'13]
 - Widely distributed; several different layers
 - Usually a user-facing '**frontend**' (via web protocols) and an internal '**backend**' (e.g., for storage)
 - Multiple components with complex interactions
- This is too much work to build by yourself!
 - Hence, I've saved this for the final project, which we'll do in teams

Final project: PennCloud



- Goal: Implement a small clone of 'Google Apps'
 - Just two apps: **Webmail** and **storage**
 - No need to replicate the fancy optics (just some of the key features)
 - Should include a **web frontend** and a **key-value store backend**
 - Should be done in teams of four
- Can apply many of the things we've discussed!
 - RPCs, fault tolerance, replication, client/server, synchronization, socket programming, distributed coordination, ...
 - Also: Nice addition to your portfolio! (e.g., for interviews)

Examples from previous years

Storage

The first screenshot shows a login page with fields for 'username' and 'password', and a 'SUBMIT' button. Below it is a note for new users: 'New user? Register [here](#)'.

The second screenshot shows the main storage interface with a sidebar menu (Mails, Drive, Admin, Game) and a central area for managing files. It displays a 'CURRENT DIRECTORY: root' with folders 'ictures', 'secrets', and 'documents'. A file named 'stora.org' is selected, showing options to 'Rename' or 'Delete'. A confirmation dialog asks, 'Are you sure you want to delete this file?' with 'Delete' and 'Move' buttons.

The third screenshot shows a game lobby screen. It displays a unique ID: '1529278520273215' and a 'START GAME' button. A note says: 'Please share this unique ID with player two. Note: Please don't click start button before player two joins.'

Game

The first screenshot shows a dashboard with a table listing various services (Mails, Drive, Admin, Game) with columns for Server Port, Active, Current Load, Tablet Information, and Disable/Enable.

The second screenshot shows an email interface with a compose button and three outgoing messages:

- Subject: FWD: Tickets for Penn Relay? From bob@PennCloud on Tue May 8 14:23:44 2018
- Subject: CIS505 From eve@PennCloud on Tue May 8 14:21:47 2018
- Subject: Hi Alice From bob@PennCloud on Tue May 8 14:19:09 2018

The third screenshot shows a game board interface with a 3x3 grid. The top row contains a green circle and an empty square. The middle row contains a green circle and a red cross. The bottom row contains a red cross and a green circle. A 'Refresh' button is visible above the board.

Admin

The first screenshot shows a dashboard with a table listing various items (Mails, Drive, Admin, Game) with columns for User, Column, and Value.

The second screenshot shows an email interface with a compose button and three incoming messages:

- Subject: CIS505 From bob@PennCloud on Tue May 8 14:23:44 2018
- Subject: CIS505 From eve@PennCloud on Tue May 8 14:21:47 2018
- Subject: Hi Alice From bob@PennCloud on Tue May 8 14:19:09 2018

The third screenshot shows a game board interface with a 3x3 grid. The top row contains a green circle and an empty square. The middle row contains a green circle and a red cross. The bottom row contains a red cross and a green circle. A 'Refresh' button is visible above the board.

Webmail

20

Examples from previous years

PennCloud

Increase your productivity and safely store what matters most with PennCloud.



PennMail



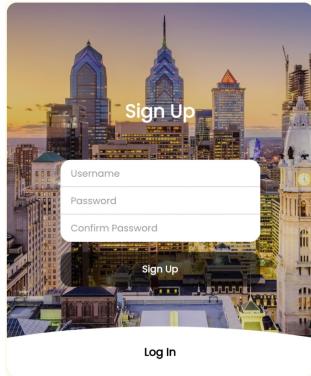
PennDrive



PennDiscuss



PennChat



PennCloud

Compose

Inbox

Reply

Forward

Delete

From: Amit@penncloud.com
Subject: Art exhibition at the Met. Let's go!
Date: Tue Dec 22 02:32:59 2020

Hey Bharath,

I read this article recently about a fantastic Michelangelo exhibit at the Met. Shall we go?

Here's the link to the article I read about it: <https://www.nytimes.com/2017/11/09/arts/design/michelangelo-review-metropolitan-museum-of-art-carmen-bambach.html>

Let me know if you're free. See you!

Regards,
Amit Lohe

Webmail

PennCloud

Change Password

Logout

Parent Directory

Select File... Upload File

New Directory Name... Create Directory

~/Penn/

DistributedSystems.pdf	New Name...	Rename	New Absolute Path...	Move	
gift_ideas.docx	New Name...	Rename	New Absolute Path...	Move	
Homework.txt	New Name...	Rename	New Absolute Path...	Move	
ML_Data	New Name...	Rename	New Absolute Path...	Move	
Practice_Exams	New Name...	Rename	New Absolute Path...	Move	
Rec Letters	New Name...	Rename	New Absolute Path...	Move	

Storage

PennChat

Bharath: Hey guys!

Bharath: What are your plans for winter break?

Prasanna: Thinking of skydiving!

Bharath: Indoor or outdoor? I've only been indoor...

Prasanna: Outdoor! It's way more fun and the view is worth it...

Amit: I've been to both - I loved outdoor, it's way more scary

Amit: Where are you planning to go? Near Philly?

Prasanna: Yes!

Chat

PennDiscuss

Liana: Online Board Game Suggestions

Liana: Hi everyone! I was wondering if anyone could suggest some online board games to help stay connected with friends during quarantine?

Amit: Many different college groups on Facebook have posted lists of online board games... I'll email one to you

Amit: Just emailed one to you!

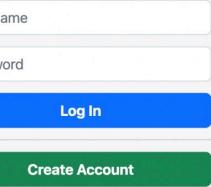
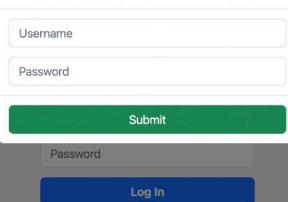
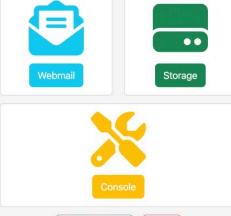
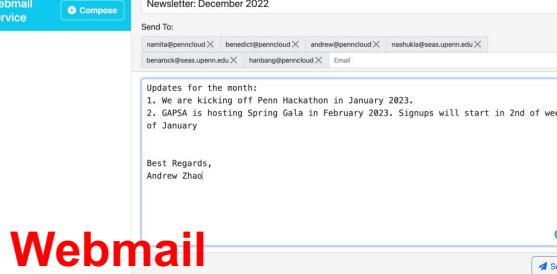
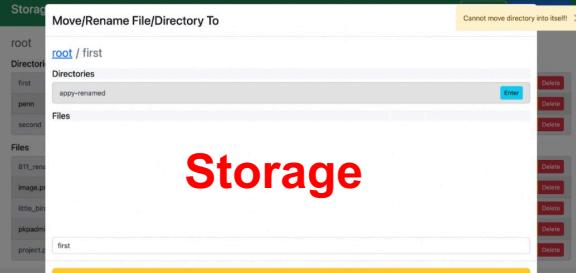
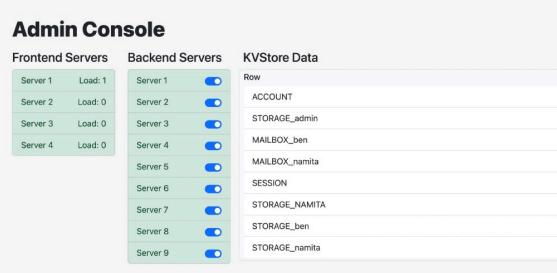
Prasanna: I think Bharath and his friend created an online board game called Codewords: www.codewordsgame.com

Bharath: Thanks for sharing, Prasanna!

Discussion board

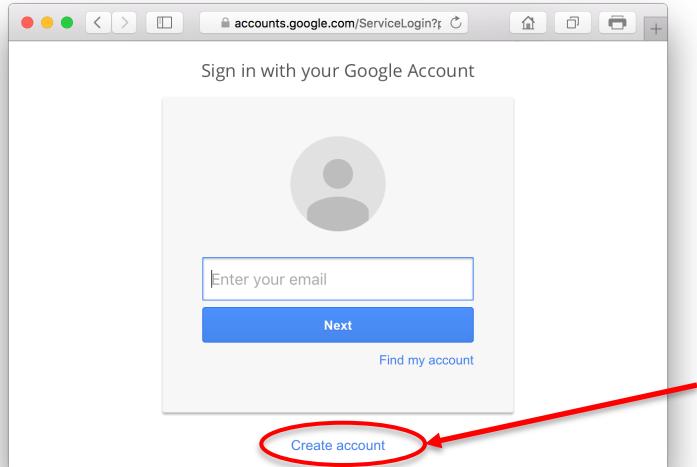
Examples from previous years

Log in

<p>PennCloud</p>  <p>Username Password Log In Create Account</p>	<p>Register Account</p>  <p>Username Password Submit Password Log In Create Account</p>	<p>PennCloud Welcome, admin!</p>  <p>Webmail Storage Console Change Password Logout</p>																																																												
<p>Webmail Service</p>  <p>Do you wanna watch Avatar? From: namita <namita@penncloud> To: ben <ben@penncloud> Date: Fri Dec 16 2022 15:59:46 GMT-0500 (Eastern Standard Time)</p> <p>BEN 12/16/2022, 3:43:01 PM From namita <namita@penncloud></p> <p>I have free ticks</p> <p>Compose</p> <p>Reply Forward Delete</p> <p>PennCloud © 2022 Created by Andrew, Benedict, Hanbang & Namita</p>	<p>Webmail Service</p>  <p>Newsletter: December 2022 Send To: namita@penncloud X benedict@penncloud X andrew@penncloud X nashika@seas.upenn.edu X benaroc@seas.upenn.edu X hanbang@penncloud X Email</p> <p>Updates for the month: 1. We are kicking off Penn Hackathon in January 2023. 2. GAPSA is hosting Spring Gala in February 2023. Signups will start in 2nd week of January</p> <p>Best Regards, Andrew Zhao</p> <p>Compose</p> <p>Send</p> <p>PennCloud © 2022 Created by Andrew, Benedict, Hanbang & Namita</p>	<p>Storage Service</p>  <p>root Directories first penn second Files 811_renamed.pdf image.png little_birds_swimming_on_muddy_pond_in_nature_6892038.zip pkpadmin+1008-4741-1-CE.pdf project.pdf</p> <p>Enter Move Delete Enter Move Delete Enter Move Delete Download Move Delete Download Move Delete Download Move Delete Download Move Delete Download Move Delete</p> <p>PennCloud © 2022 Created by Andrew, Benedict, Hanbang & Namita</p>																																																												
<p>Storage</p>  <p>Move/Rename File/Directory To Cannot move directory into itself! root / first Directories appy-renamed Files 811_renamed image.png little_birds_swimming_on_muddy_pond_in_nature_6892038.zip pkpadmin+1008-4741-1-CE.pdf project.pdf first</p> <p>Move</p>	<p>Admin Console</p>  <p>Frontend Servers Backend Servers KVStore Data</p> <table border="1"> <thead> <tr> <th>Frontend Servers</th> <th>Backend Servers</th> <th>KVStore Data</th> </tr> </thead> <tbody> <tr> <td>Server 1 Load: 1</td> <td>Server 1</td> <td>Row</td> </tr> <tr> <td>Server 2 Load: 0</td> <td>Server 2</td> <td>ACCOUNT</td> </tr> <tr> <td>Server 3 Load: 0</td> <td>Server 3</td> <td>MAILBOX_admin</td> </tr> <tr> <td>Server 4 Load: 0</td> <td>Server 4</td> <td>STORAGE_admin</td> </tr> <tr> <td></td> <td>Server 5</td> <td>STORAGE_andrew</td> </tr> <tr> <td></td> <td>Server 6</td> <td>STORAGE_han</td> </tr> <tr> <td></td> <td>Server 7</td> <td>MAILBOX Ben</td> </tr> <tr> <td></td> <td>Server 8</td> <td>SESSION</td> </tr> <tr> <td></td> <td>Server 9</td> <td>STORAGE_namita</td> </tr> </tbody> </table> <p>Move</p>	Frontend Servers	Backend Servers	KVStore Data	Server 1 Load: 1	Server 1	Row	Server 2 Load: 0	Server 2	ACCOUNT	Server 3 Load: 0	Server 3	MAILBOX_admin	Server 4 Load: 0	Server 4	STORAGE_admin		Server 5	STORAGE_andrew		Server 6	STORAGE_han		Server 7	MAILBOX Ben		Server 8	SESSION		Server 9	STORAGE_namita	<p>KVStore Data</p> <table border="1"> <thead> <tr> <th>Row</th> <th>Column</th> <th>Data</th> </tr> </thead> <tbody> <tr> <td>ACCOUNT</td> <td>c</td> <td>%PDF-1.6 %♦♦♦♦ 3096 0 obj <</Linearized 1/L 597946/O 3098/E 186670/N 3/T 596160/H [1305 6670]>> endobj</td> </tr> <tr> <td>MAILBOX_admin</td> <td>9</td> <td><</DecodeParms<</Columns 5/Predictor 12->/Filter/FlateDecode/IID<E2361 B90419C34A82CCBBA591699A15><D53163156C7BE7488AAFA85E5D 13379>/>>>/Index[3096 311]/Info 3095</td> </tr> <tr> <td>STORAGE_admin</td> <td>8</td> <td></td> </tr> <tr> <td>STORAGE_andrew</td> <td>7</td> <td></td> </tr> <tr> <td>STORAGE_han</td> <td>6</td> <td></td> </tr> <tr> <td>MAILBOX Ben</td> <td>5</td> <td></td> </tr> <tr> <td>SESSION</td> <td>4</td> <td></td> </tr> <tr> <td>STORAGE_namita</td> <td>3</td> <td></td> </tr> <tr> <td></td> <td>2</td> <td></td> </tr> </tbody> </table>	Row	Column	Data	ACCOUNT	c	%PDF-1.6 %♦♦♦♦ 3096 0 obj <</Linearized 1/L 597946/O 3098/E 186670/N 3/T 596160/H [1305 6670]>> endobj	MAILBOX_admin	9	<</DecodeParms<</Columns 5/Predictor 12->/Filter/FlateDecode/IID<E2361 B90419C34A82CCBBA591699A15><D53163156C7BE7488AAFA85E5D 13379>/>>>/Index[3096 311]/Info 3095	STORAGE_admin	8		STORAGE_andrew	7		STORAGE_han	6		MAILBOX Ben	5		SESSION	4		STORAGE_namita	3			2	
Frontend Servers	Backend Servers	KVStore Data																																																												
Server 1 Load: 1	Server 1	Row																																																												
Server 2 Load: 0	Server 2	ACCOUNT																																																												
Server 3 Load: 0	Server 3	MAILBOX_admin																																																												
Server 4 Load: 0	Server 4	STORAGE_admin																																																												
	Server 5	STORAGE_andrew																																																												
	Server 6	STORAGE_han																																																												
	Server 7	MAILBOX Ben																																																												
	Server 8	SESSION																																																												
	Server 9	STORAGE_namita																																																												
Row	Column	Data																																																												
ACCOUNT	c	%PDF-1.6 %♦♦♦♦ 3096 0 obj <</Linearized 1/L 597946/O 3098/E 186670/N 3/T 596160/H [1305 6670]>> endobj																																																												
MAILBOX_admin	9	<</DecodeParms<</Columns 5/Predictor 12->/Filter/FlateDecode/IID<E2361 B90419C34A82CCBBA591699A15><D53163156C7BE7488AAFA85E5D 13379>/>>>/Index[3096 311]/Info 3095																																																												
STORAGE_admin	8																																																													
STORAGE_andrew	7																																																													
STORAGE_han	6																																																													
MAILBOX Ben	5																																																													
SESSION	4																																																													
STORAGE_namita	3																																																													
	2																																																													

Admin

How users interact with the project

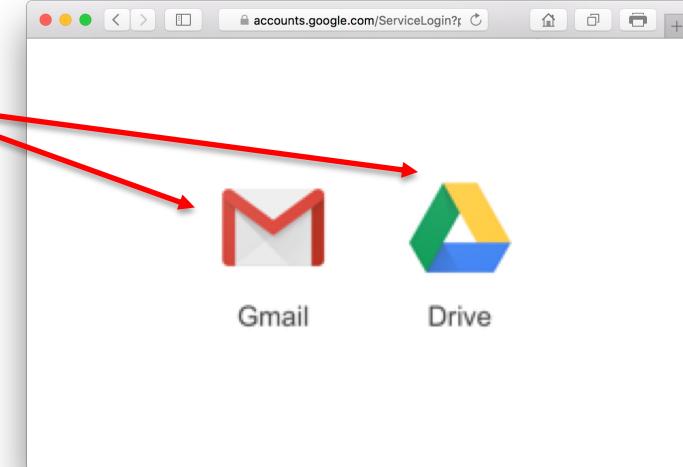


Login page: Enter username and password, or sign up for new account

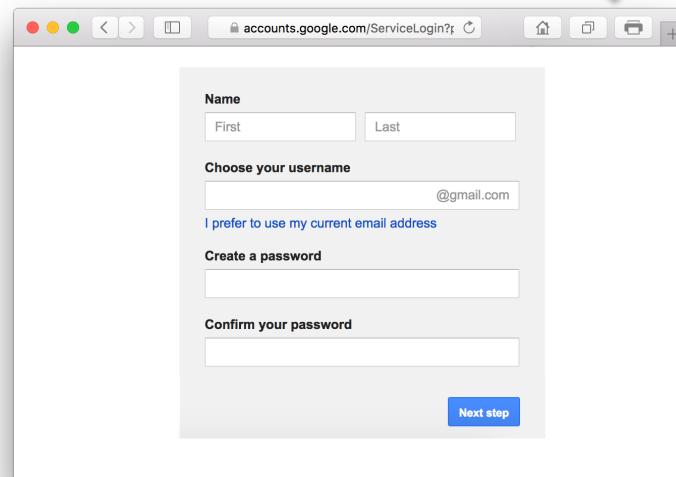
Links to
webmail
& storage



Sign up
link



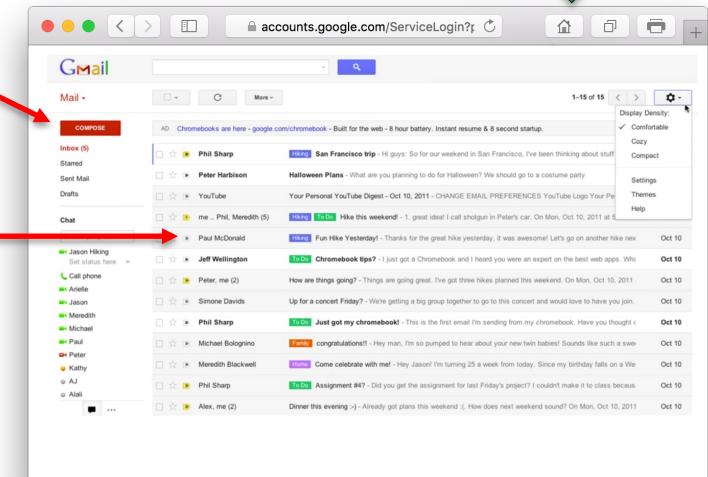
Menu page: Go to the webmail service or to the storage service



Signup page: Enter information for a new account

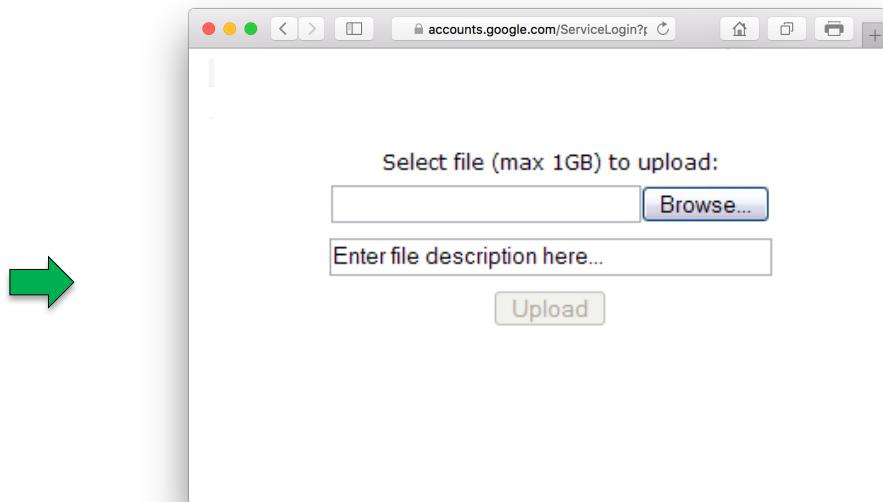
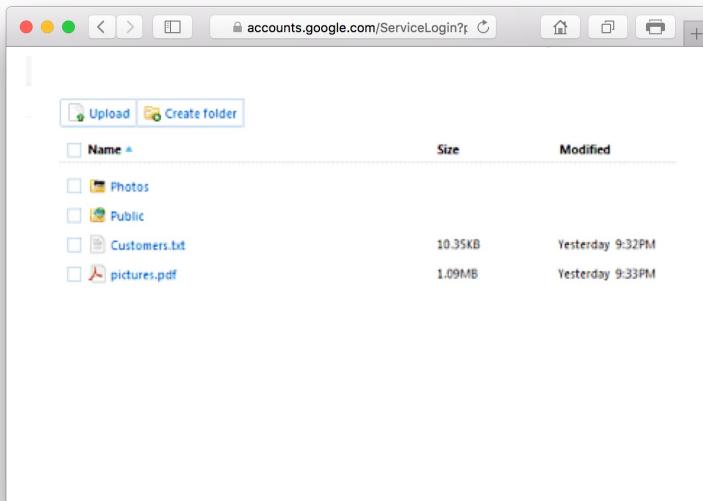
Compose
button

List of
emails

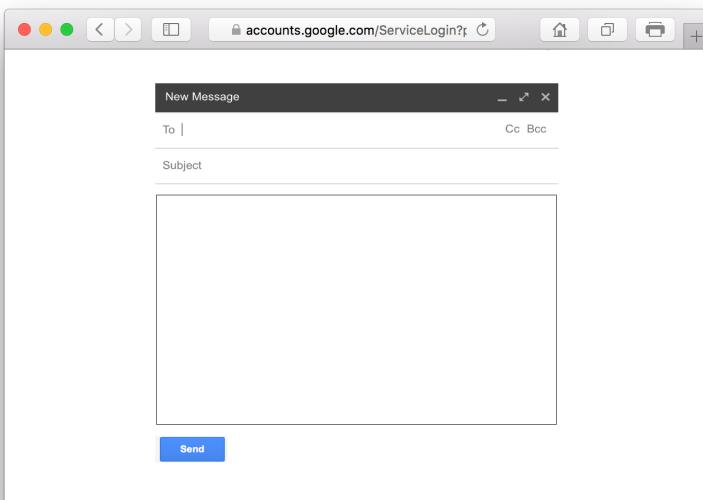


Webmail page: Shows list of emails in inbox & a "compose" button

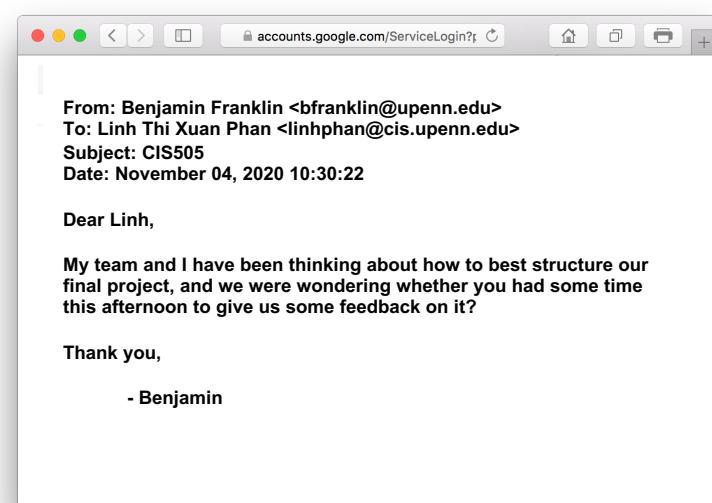
How users interact with the project



Directory page: Displays the files the user has currently stored in the system

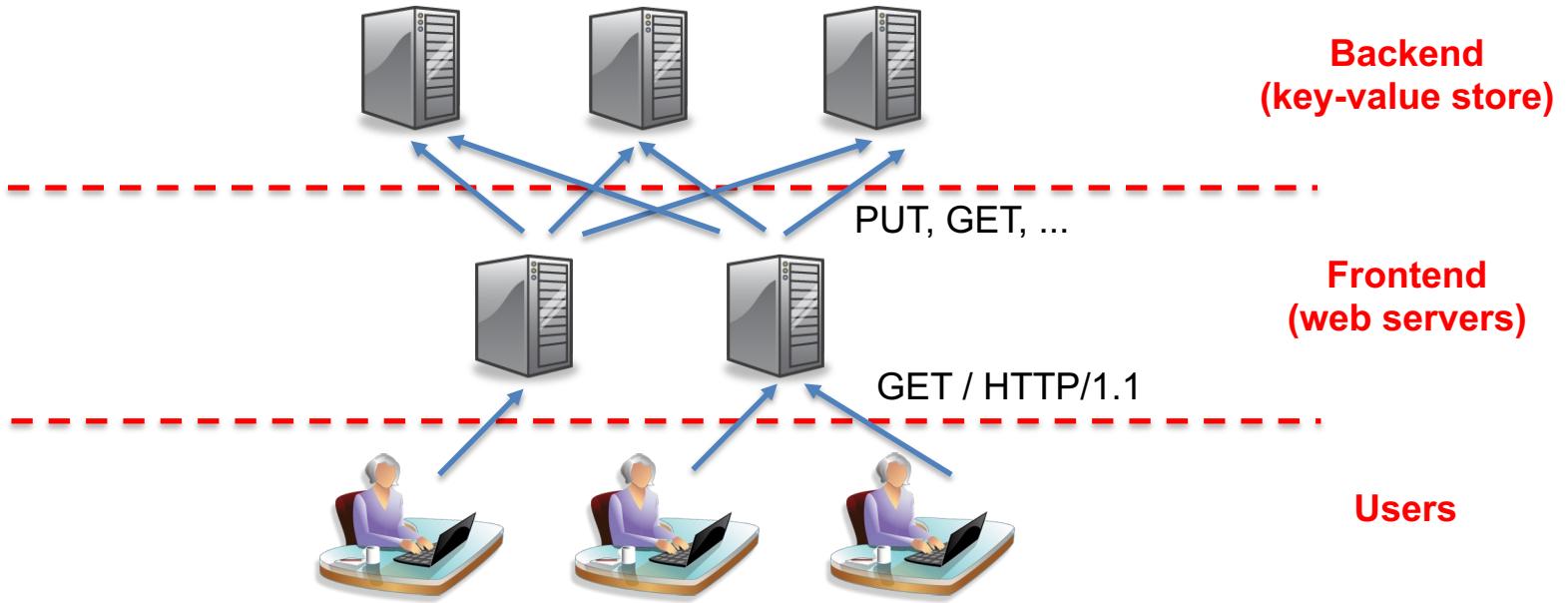


Compose page: Lets the user enter a new email to be sent



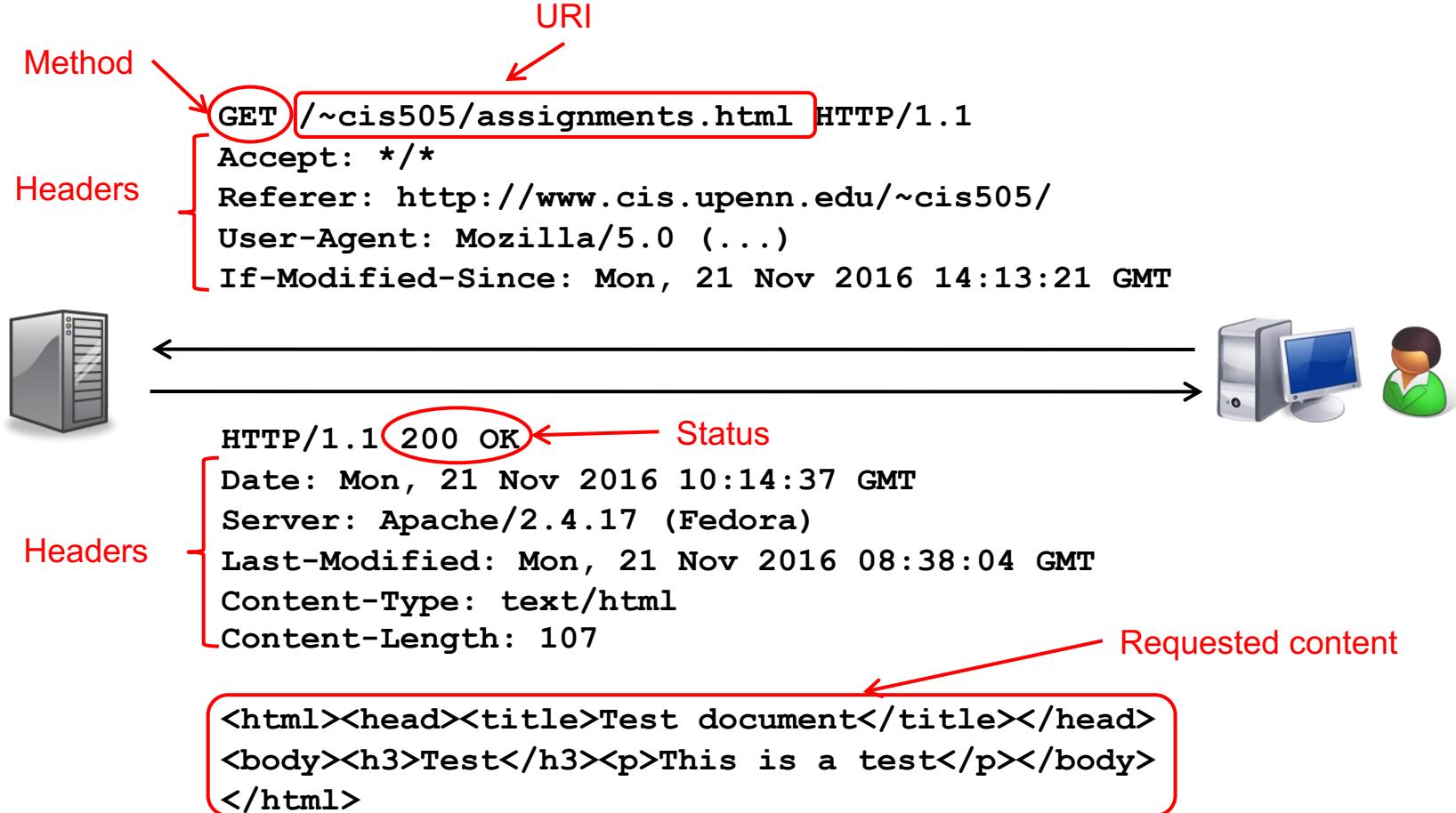
Email page: Shows an individual email

Major components



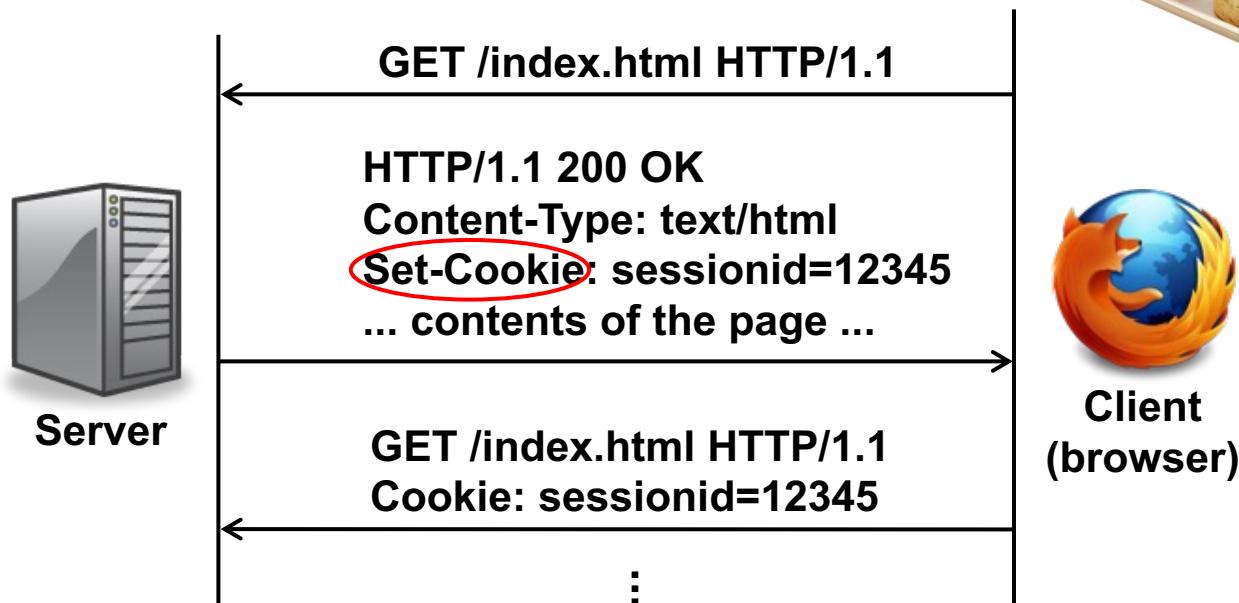
- **Frontend tier**: Several web server instances
- **Backend tier**: Several key-value store nodes
- User can connect to any front-end server
 - All the state is supposed to be stored in the key-value store!

HTTP in one slide



- Very similar to the servers you wrote for HW2
 - Text-based requests and responses; lots of headers with extra information, but simple implementations can omit most of them
 - For more details: <https://www.jmarshall.com/easy/http/>

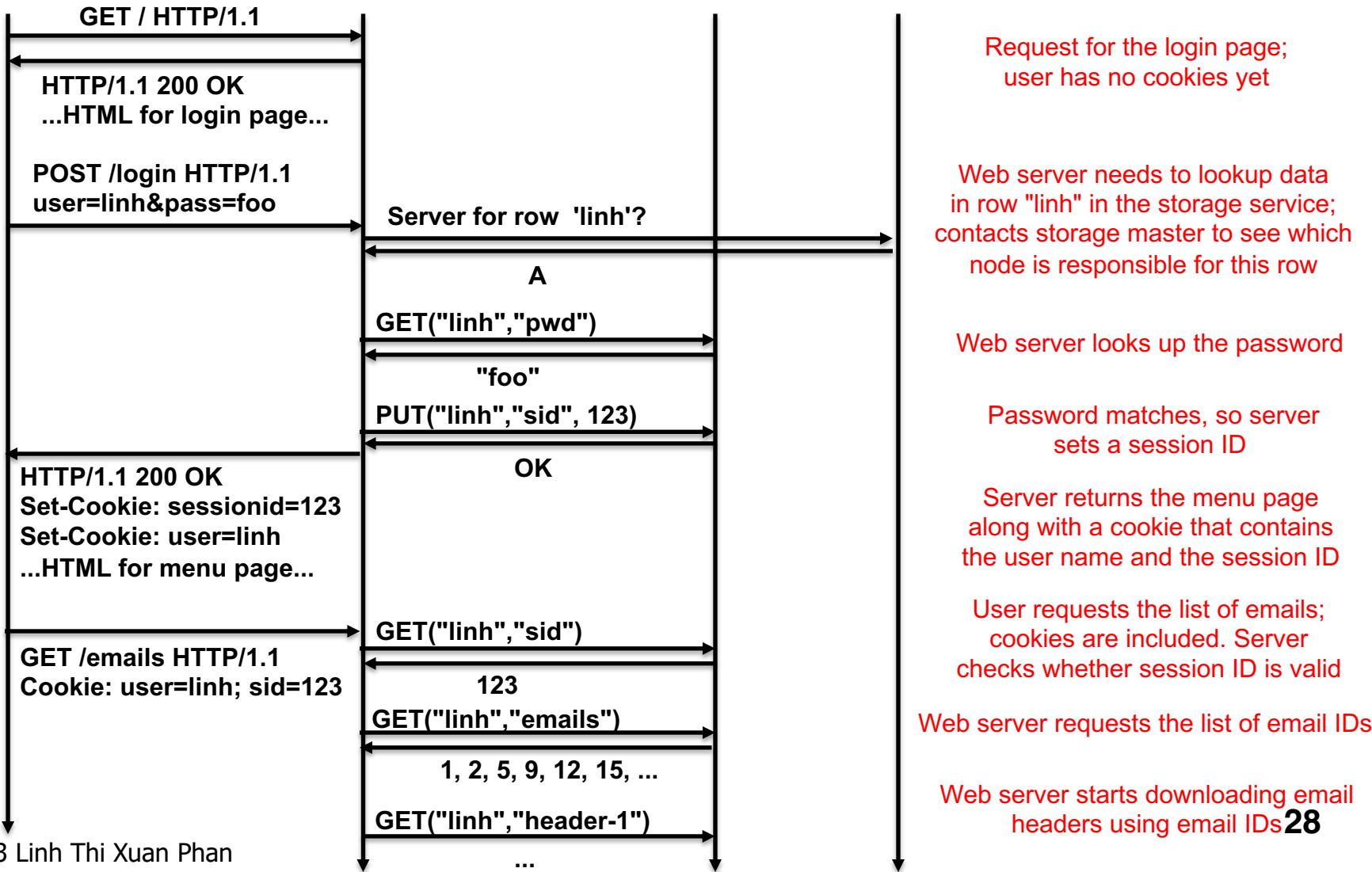
HTTP cookies



- What is a **cookie**?

- A set of key-value pairs that a web site can store in your browser (example: 'sessionid=12345')
- Created with a Set-Cookie header in the HTTP response
- Browser sends the cookie in all subsequent requests to the same web site until it expires

Example session



How fancy does it have to be?

- There is no need to replicate Gmail visuals etc.
 - Very basic HTML (unordered lists etc.) are sufficient
- However, the project is open-ended!
 - We will liberally award extra credit for extra features
 - Please get the basics working first and only then start on EC tasks!
- The project will be graded on
 - Functionality: Are all the required features there? Do they work?
 - Design: Is the system well-designed? (Bottlenecks? Fault tolerance? Recovery? Scalability? Etc.)
 - Usability: Good user interface, etc.

Resources

- Storage backend
 - Bigtable paper (research.google.com/archive/bigtable-osdi06.pdf)
- Web frontend
 - HTTP Made Really Easy (<https://www.jmarshall.com/easy/http/>)
 - RFC 2616 (<https://www.ietf.org/rfc/rfc2616.txt>) – basic HTTP
 - RFC 2965 (<https://tools.ietf.org/html/rfc2965>) – cookies
- HTML
 - <http://www.w3schools.com/html/>