Team 03 - UDK

Hamza Hameed | Jan Polzer | Ron Huff | Ethan Malin | Alejandro Melgar

Synopsis

A mobile application for the University Daily Kansan newspaper.

Description

In the fall semester the UDK approached the senior design class with an opportunity. They would like one team to create a mobile application for their newspaper. We jumped on the project immediately and here we are. The final deliverable is a mobile application built for the iOS and Android operating systems. It will provide users with the news as produced by the UDK in an aesthetically pleasing, and user-friendly fashion.

Milestones

First Semester:

1. Analyze project requirements
2. Break down html article into python object
3. Set up back end server environment
4. Create frontend alpha
5. Allow communication between server and mobile client

Second Semester

1. Set up MySQL database on the final server
2. Send data ‘end to end’ from the UDK site to a phone
3. Display data on mobile app
4. Bring about realization of front-end design
5. Publish mobile application on app store

Budget

Estimated cost

$124.00 + $99/year to host the app on both app stores.
This cost will be the responsibility of the UDK.

Hardware, software, and/or computing resources

N/A

Vendor

The app will be hosted on the Google Play store and the Apple App store

Special training

N/A

When they will be required

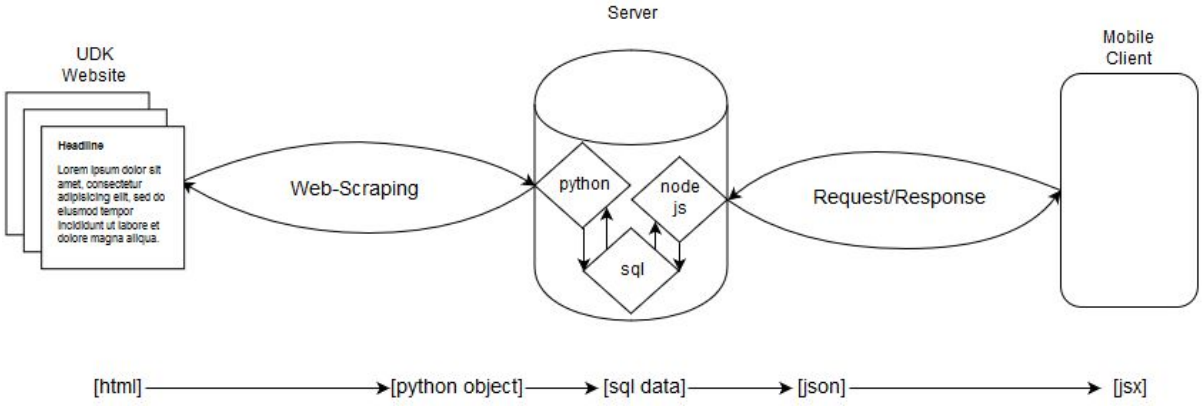
3/19/2019

Work Plan

Ron Huff (Back-end web scraping and storage)
Ethan Malin (Back-end support and client-server communications support)
Jan Polzer (Client-server communications)
Hamza Hameed (Front end design)
Alejandro Melgar (Front end design)

Final Project Design

The project is best visualized as a data pipeline, with various transformations applied along the pipeline. The media starts hosted on the Blox CMS servers as html. Our program grabs the html and turns it into a python object. That python object is then stored in a SQL server. From the SQL server it it JSONified by our node, and sent through the network to the mobile device, where it is finally rendered as JSX (javascript markdown). A visualization can be helpful to understand.



Web-Scraping

Because BLOX CMS does not provide an avenue for it's clients to retrieve their data through a programming interface, we must instead get the data by scraping the website of the UDK and breaking down the received HTML into its' component parts.

We are using python for this portion of the application, both for its simplicity and because we have experience using it. The beautifulsoup library also has come in handy, being a python library made for navigation an html document. With this library we are able to search the incoming html for its' important content, and transform it into something more machine friendly.

The first thing that must be done is that we must enumerate the articles already created by the UDK. To do this we use the search function of the UDK's website. We search for the oldest articles and work our way forwards by using the next button. Each returned item on the search page contains an article url that we capture for later processing.

Once we have a list of articles, we process them one by one, starting from the oldest. It is possible to fail to process an article. This may happen if BLOX CMS changed the format of their webpage at some point in time. They might have started using a different class or a different format for their markdown that we are not specifically searching for. If we fail 10 articles in a row we abort this process to manually inspect the content of those pages for what has changed. On finding the mistake, we can correct our algorithm and start back up where we left off. On successfully processing an article, we store it in our server's database for later retrieval by the node server.

The other function that we use the web scraping for is to keep our database up to date. Every day we will use the search page to get the 25 most recent articles. We process the articles until we find a headline that we have already stored and at that point we know that there are no new articles.

Database

For our database we are using a mysql server for the simple reason that it works, and that we have experience with it. Designing a database for an application was a new problem to us, and we found that the question of how to organize the data is not a simple as it may seem at first. A single "story" posted by the UDK can be many things, from a single picture, to large body of text with embedded images, video, and audio.

Our column header looks like this:

ID	URL	HEADLINE	AUTHOR	DATE	MAIN_IMAGE	BODY

Some comments about each of the sections.

ID

This field is autogenerated and is used to uniquely identify each article.

HEADLINE / AUTHOR / DATE

These fields are self-explanatory, but it is worth noting that they are all required. These and the ID make up the minimal article possible.

MAIN_IMAGE

This is a link to the image that is used as the thumbnail, and at the head of the article.

BODY

The only one that really needs an explanation. The problem we had was that the article consists of text, split up into paragraphs, with images inserted randomly in between. Somehow we had to maintain the formatting across all of the data transformation. It may not be the prettiest solution, but what we came up with was to encode the entire article as a string, with special characters denoting where paragraphs begin, where images are to be inserted, and when any special formatting is called for. An example story would be

“$$$PARAGRAPH$$$Some news about the town, and what the people are doing $$$IMAGE$$$link-to-an-image-here.jpg $$$ASIDE$$$aside is a special formatting the UDK uses to highlight a quote$$$PARAGRAPH$$$ and this is the end of the article

When the body is sent to the phone, it must break down the article according to the special characters it finds within the text. Hopefully the UDK never writes $$$PARAGRAPH$$$ in one of its’ articles, although we doubt that will happen.

Node Server

The node server acts as the intermediary between the mobile client and the sql server. We chose to use node primarily so that we could use javascript on the front and back ends of the application. React native is a javascript framework, and so it is extremely easy to send JSON data back and forth between the server and the client. Node (and javascript in general) also has lots of structure in place for asynchronous operation, something that will allow us to scale our application to accommodate many users.

Our node server needs to ‘return’ one of a few things. The easiest way to see what needs to be available is by stepping through a typical run of the front end application, see the Front End section of this document to read about that.

The node server must be able to serve

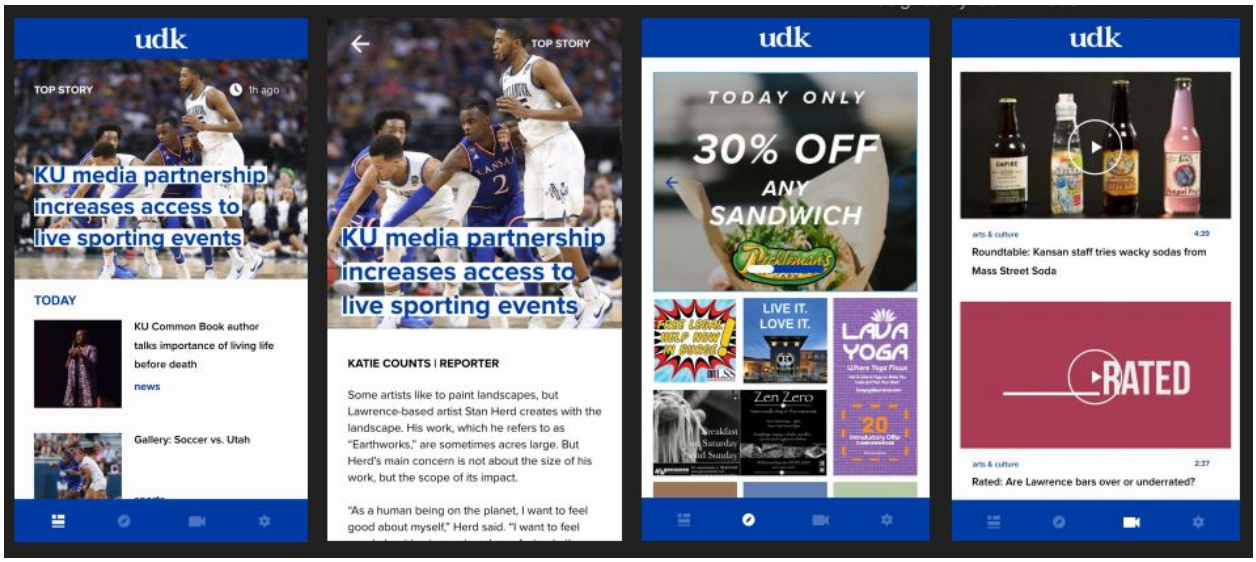
1. A list of the 25 most recent cards.
2. An entire article, requested by its’ article ID.
3. A list of 25 cards, sorted by best match to a user query.

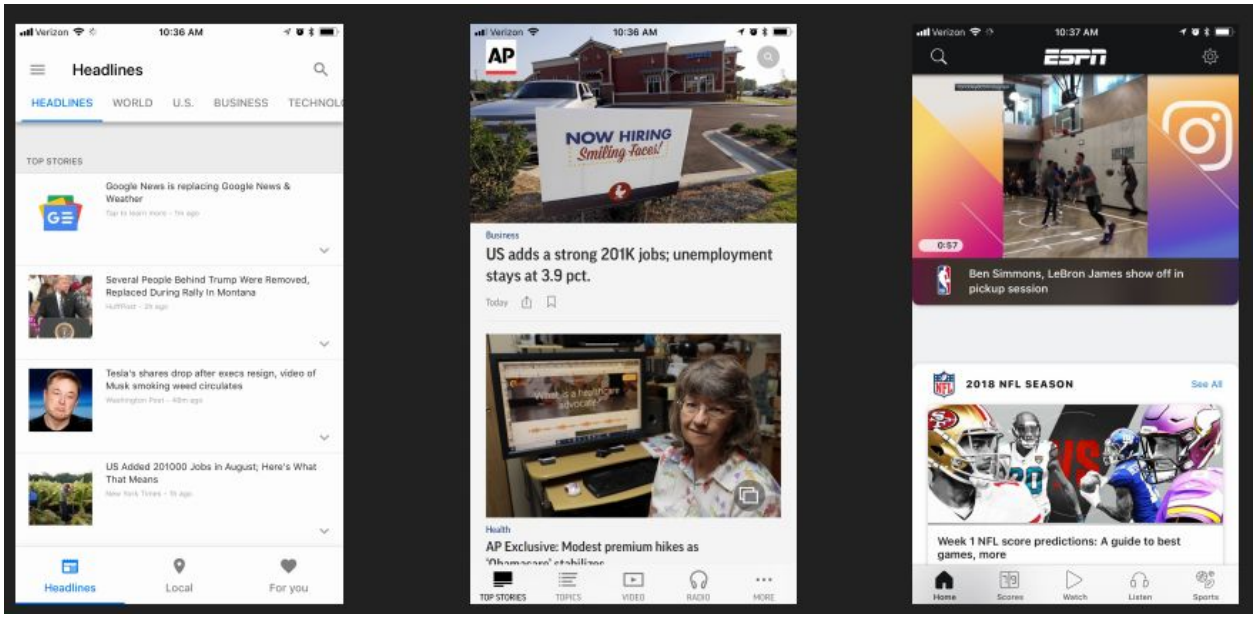
Socket Server

Second node server that will be used for notifications using sockets. It will be able to notify all connected hosts/applications to refresh when there are new articles in the database..

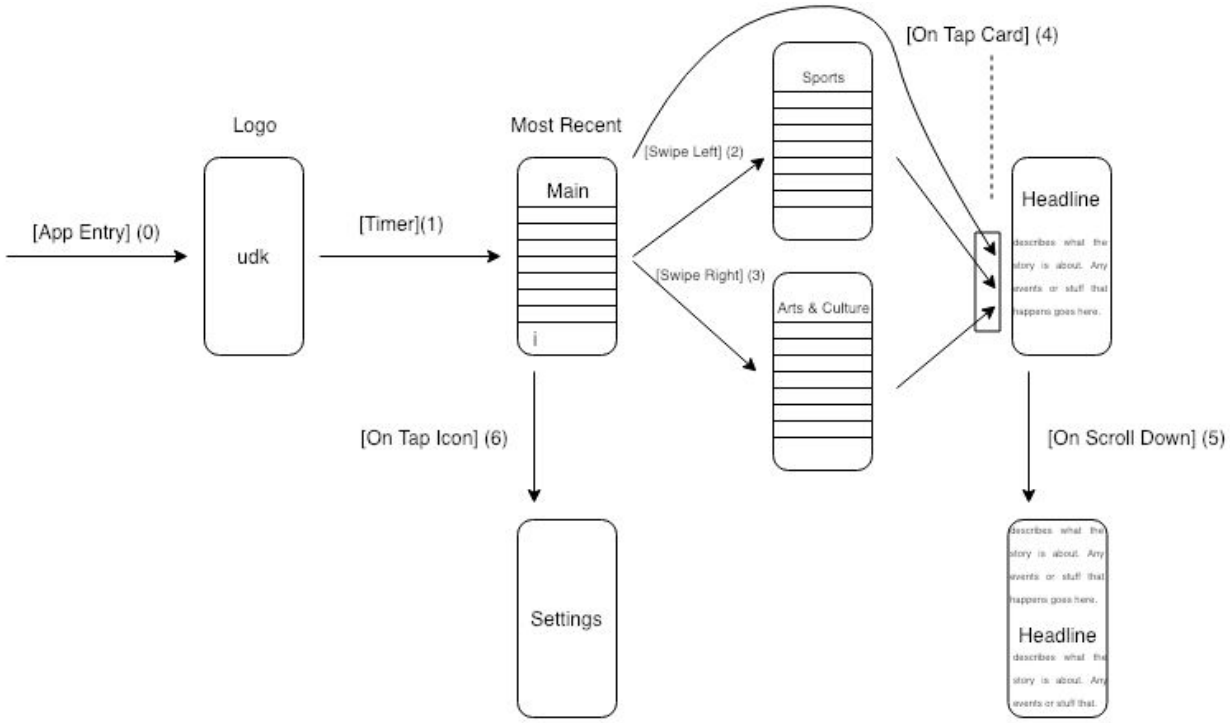
Front end

The University Daily Kansan left the design of the app up to us. We are the designing the app to be consistent with the design of the Kansan website. On top of attempting to keep consistent design we are looking to other popular news applications, such as ESPN, The Associated Press to name just a few, which upload articles for inspiration. Below are some images which will be used as a design reference for the final version of the UDK App.


udk
TOP STORY
1h ago
KU media partnership increases access to live sporting events
TODAY
KU Common Book author talks importance of living life before death
news
Gallery: Soccer vs. Utah
TOP STORY
KU media partnership increases access to live sporting events
KATIE COUNTS | REPORTER
Some artists like to paint landscapes, but Lawrence-based artist Stan Herd creates with the landscape. His work, which he refers to as "Earthworks," are sometimes acres large. But Herd's main concern is not about the size of his work, but the scope of its impact.
"As a human being on the planet, I want to feel good about myself," Herd said. "I want to feel
udk
TODAY ONLY
30% OFF
ANY
SANDWICH
LIVE IT. LOVE IT.
LAVA YOGA
Zen Zero
20
udk
arts & culture
4:39
Roundtable: Kansan staff tries wacky sodas from Mass Street Soda
RATED
arts & culture
2:37
Rated: Are Lawrence bars over or underrated?


Verizon
10:36 AM
Headlines
HEADLINES
WORLD
U.S.
BUSINESS
TOP STORIES
Google News is replacing Google News & Weather
Several People Behind Trump Were Removed, Replaced During Rally In Montana
Tesla's shares drop after execs resign, video of Musk smoking weed circulates
US Added 201000 Jobs In August; Here's What That Means
Headlines
Local
For you
AP
NOW HIRING
Smiling Faces!
Business
US adds a strong 201K jobs; unemployment stays at 3.9 pct.
Today
Health
AP Exclusive: Modest premium hikes as
TOP STORIES
TOPICS
VIDEO
RADIO
MORE
Verizon
10:37 AM
ESPN
0:57
Ben Simmons, LeBron James show off in pickup session
2018 NFL SEASON
See All
Week 1 NFL score predictions: A guide to best games, more
Home
Scores
Watch
Listen
Sports

The structure of the front end is best described visually.



This picture shows the flow of the application. (0) The app starts and the user is presented with a brief logo screen. (1) After a timeout, the main screen is loaded. The main screen consists of a list of “cards” or representations of an article. This is a headline, author, and optionally an image. The first list presented is the most recent list. (2) If the user swipes left, the screen will transition into a similar list, but sports related. (3) If the user swipes right, they will transition into the arts and culture section. From (2) or (3) the user can swipe right or left again to the logical next screen. (4) If the user taps a single card they are presented with the article in its entirety. (5) If they scroll down while reading the article, they can continue to scroll and get to the next article as determined by the previous list they were looking at. (6) From the main screen, the user can press a small icon to go to a settings and info screen.

Not included in this flowchart is the search screen. From pressing an icon on the main screen, the user will be presented with a text box, where they can search the database and have results presented in a list similar to the main screen.

Design Constraints

The first, and largest, constraint we faced was the use of BLOX CMS by the UDK. BLOX, as previously mentioned, does not allow programmatic access to their users’ uploaded content, and so we are forced to scrape the website to get the old articles. The only other option for us was to have the UDK stop using BLOX CMS (and pay a fee to get a one time export of all

of their previous content) and provide to them a content management system of our own making. This would have increased the scope of our project tenfold, and so we settled for the inelegant solution of scraping the website for data.

Within the server itself we were free to make our own decisions about what programs to use and how to use them.

On the front end we were constrained by the client requesting an iOS and Android application. Because of this we chose to use React Native. If we were making an application for only one operating system, the obvious choice would have been to use native code, as this would allow us flexibility and power that React Native can only approximate. But because of the time constraint, and because we lacked expertise on building native applications, we chose to go with the React framework.

Gantt Chart

	Start Date	End Date	Timeline	Status
UDK Mobile App	9-15-2018	5-10-2019		
First meeting with UDK	9-15-2018	9-21-2018		Completed
Get familiar with backend and software	9-22-2018	10-26-2018		Completed
Define structure of functioning App	9-29-2018	10-5-2018		Completed
Use diagrams created	10-6-2018	10-12-2018		Completed
Initial design completed	10-13-2018	10-26-2018		Completed
Test set up a server and backend	10-20-2018	10-26-2018		Completed
Create alpha build	10-27-2018	11-9-2018		Completed
Figure out data parsing, meet with UDK	11-3-2018	11-16-2018		Completed
Firure out storing data in database	11-10-2018	11-23-2018		Completed
Resolve error logging and data requests	11-17-2018	12-6-2018		Completed
Winter break	12-7-2018	1-21-2019		Completed
Reevaluate milestones and refine features	1-22-2019	2-1-2019		Completed
Start beta build	1-26-2019	2-8-2019		In progress
Update error logging, add socket server	2-4-2019	2-15-2019		In progress
Add documentation/commenting	2-11-2019	2-22-2019		Planned
Add push notifications	2-16-2019	3-1-2019		Planned
Add advertising	2-25-2019	3-8-2019		Planned
Add KUJH live streaming	3-4-2019	3-19-2019		Planned
Review with UDK	3-18-2019	3-22-2019		Planned
Add last features / fix bugs	3-18-2019	4-26-2019		Planned
Polish final build and release to download	4-20-2019	5-3-2019		Planned

Ethical Issues

The obvious ethical issue here is in the scraping of the web for the articles. BLOX CMS clearly does not want their users to have free and open access to the content they upload, and so if they heard about us scraping their site for every single article, they would probably not be happy. Our argument is twofold. First, we are working for the UDK, the original creators of the content. If anybody has true domain over how the articles are used and seen, it is the UDK. Secondly, anybody with too much time and lots of paper could do exactly what we are doing by hand. They could start from the oldest article and transcribe them however they see fit for later use. Through this argument we do no wrong.

Change Log

We have planned on parsing RSS feeds to collect UDK articles from their website, but then we discovered a better way of doing it. We are using beautifulsoup combined with the website's search function for that purpose.