CyRIS (May 14-04)
Design Document

Team Members:

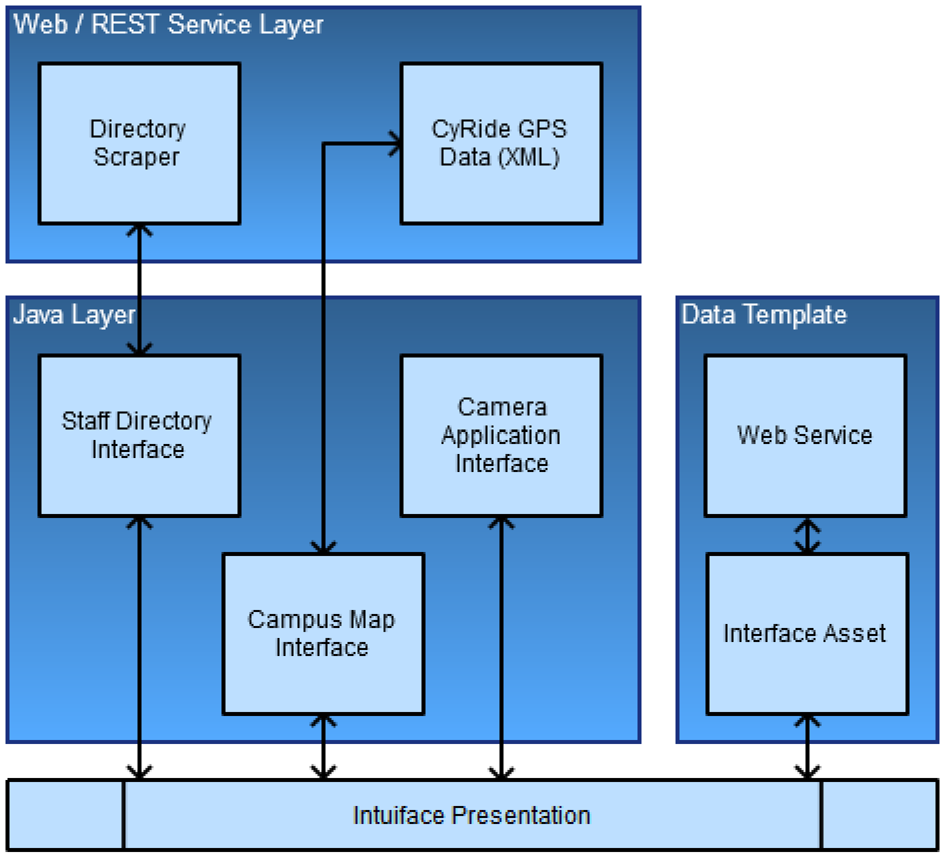
Nathan Clague
Michael Krantz
Zachary Patzwald
Maxwell Philips
Jake Roman
Micah Stevenson
David Vriezen

Advisors:

Dr. Manimaran Govindarasu
Brock Ascher

I. Table of Contents

II. High Level Architecture Diagram



The three main abstraction layers in our project are the existing Intuiface presentation (which will be extended to include our feeds application), the Java layer containing code written using the MT4J framework, and the web service layer providing communication with various endpoints so we can pull external data into our project.

Intuiface's native Interface Asset element will be utilized to display web data pulled from specific ISU-related feeds.

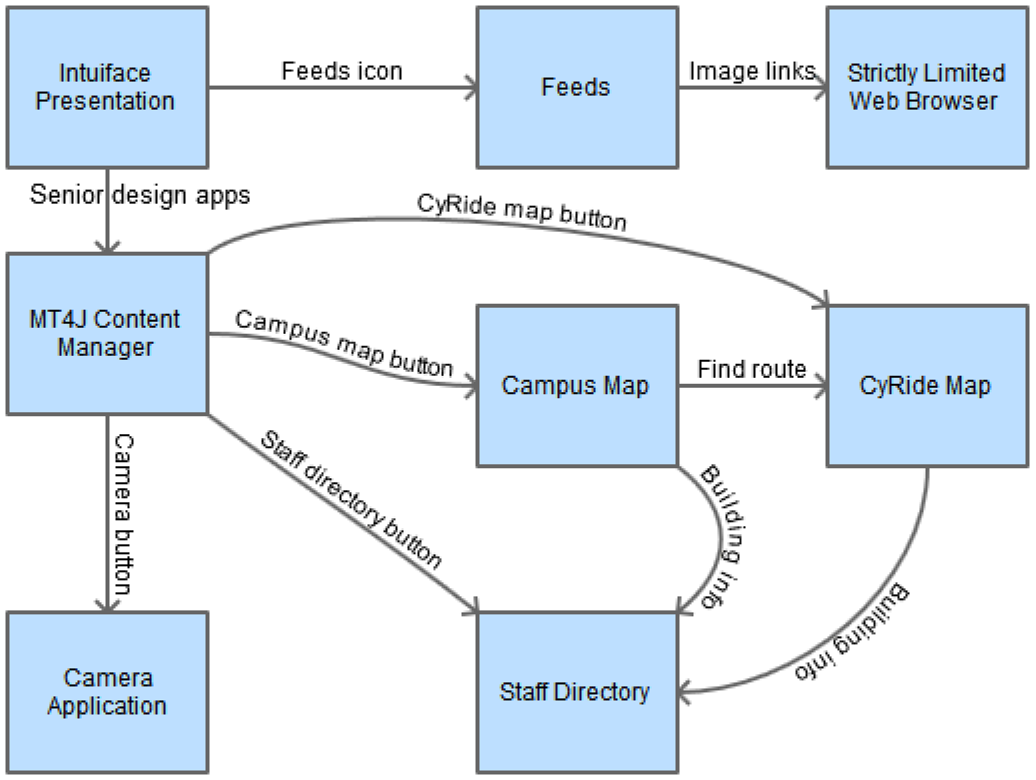
Our Java layer will contain frontend code supporting multi touch gestures that will allow users to interact with our applications in a meaningful way.

The web services we leverage will return staff directory information or NextBus XML for injection into our project.

III. User Interface Design Layout

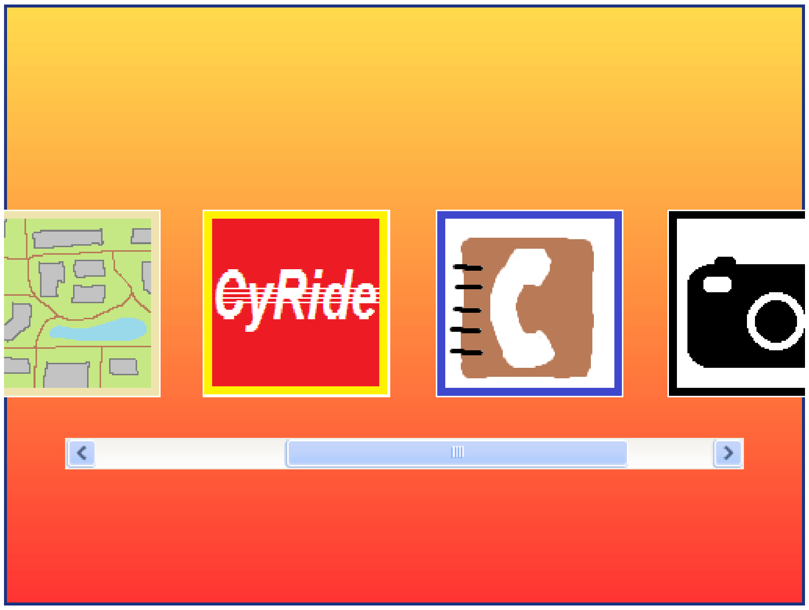
User interface design is a large aspect of this project as we want our applications to be visually appealing in order to draw users in. The following diagrams show the proposed UI flow in addition to mock up views of our applications.

UI Flow Diagram



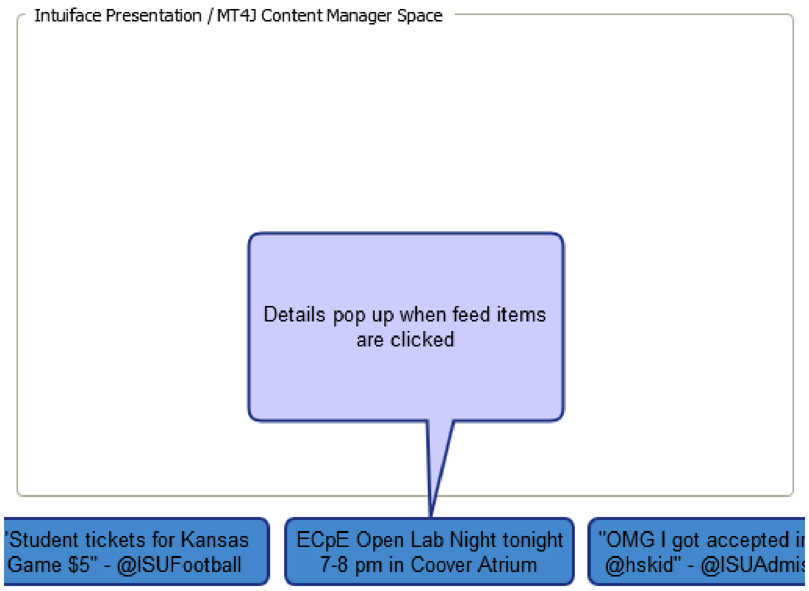
Here, blue boxes represent UI views and the arrows between them represent actions a user may take to move between the various areas of our project.

Content Manager Mockup



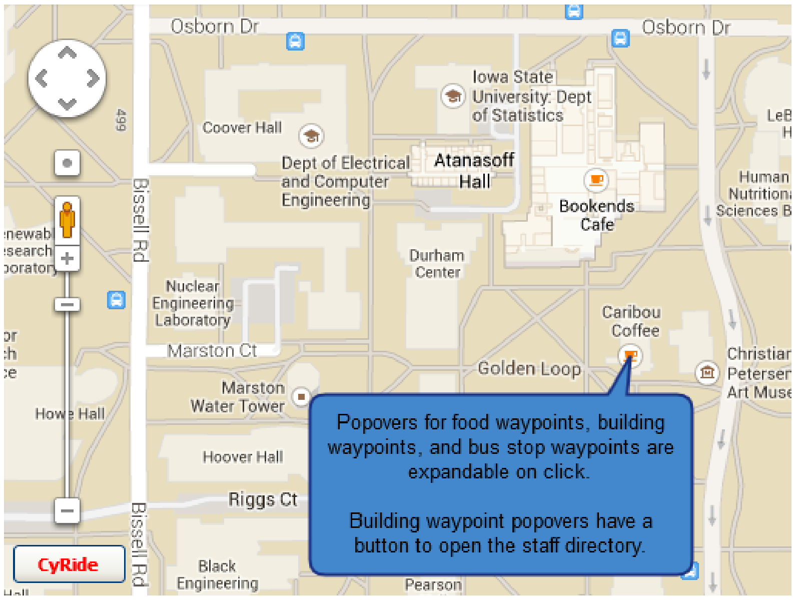
The content manager will have a scrollable list of application icons. When a user selects an application, the content manager will spawn an interactive window containing the application. The window can be resized, rotated, and dragged across the screen.

Feeds Application Mockup


Intuiface Presentation / MT4J Content Manager Space
Details pop up when feed items are clicked
'Student tickets for Kansas Game $5" - @ISUFootball
ECpE Open Lab Night tonight 7-8 pm in Coover Atrium
"OMG I got accepted i
@hskid" - @ISUAdmi

The blue spaces will crawl across the bottom of the screen, informing users about current events relating to the university and ECpE department. Users can expand items they want to know more about to get further details.

Campus Map Application Mockup

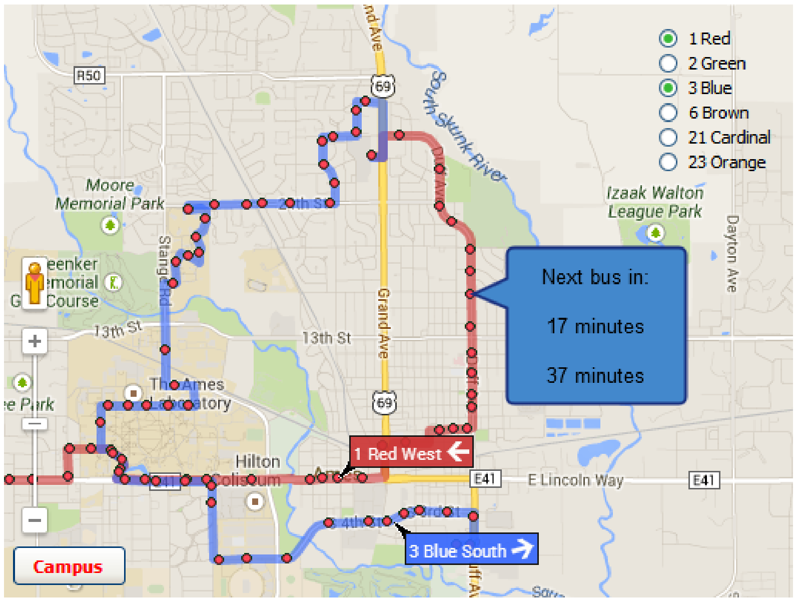


Popovers for food waypoints, building waypoints, and bus stop waypoints are expandable on click.

Building waypoint popovers have a button to open the staff directory.

The campus map application will be realized via the Google Maps API. As such, native Google Maps controls will be present along with expandable map waypoints our team will embed into the map. Note that CyRide information does not appear by default, and must be added to the map by clicking the CyRide button.

CyRide Map Application Mockup



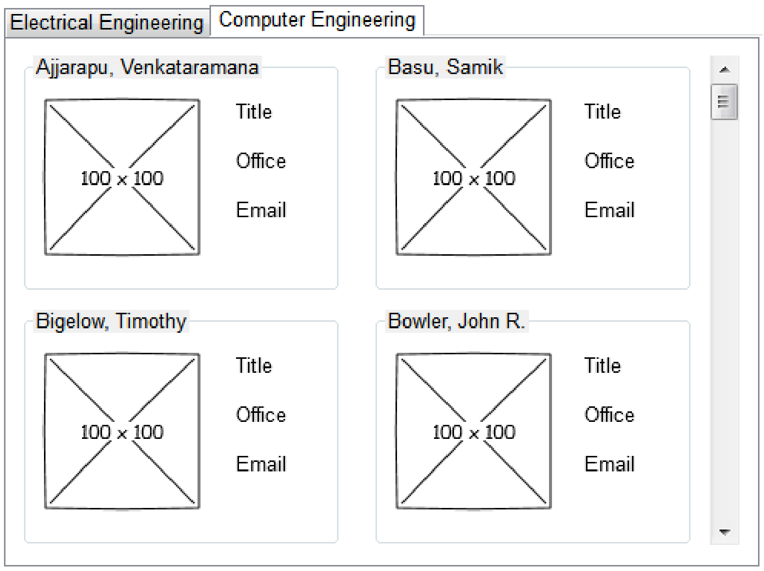
Next bus in:

17 minutes

37 minutes

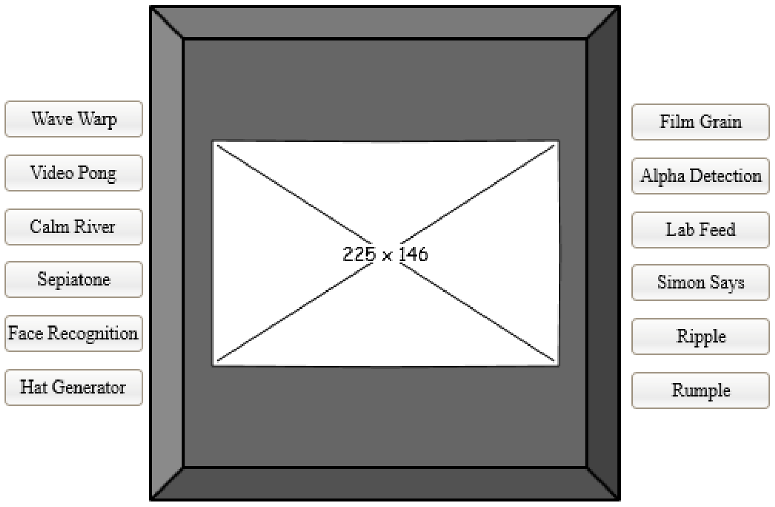
Once a user switches the map to CyRide mode, he or she can add bus routes to the display and expand time point stops to see when the next bus will arrive. Not every stop is a time point, those that are and are not have already been defined by NextBus. Additionally, a user can input a start and end point and ask the map which bus route will get them there the fastest.

Staff Directory Application Mockup



The staff directory
application will be built as a
baseline with room for expansion.
It is launchable from the home screen or from the

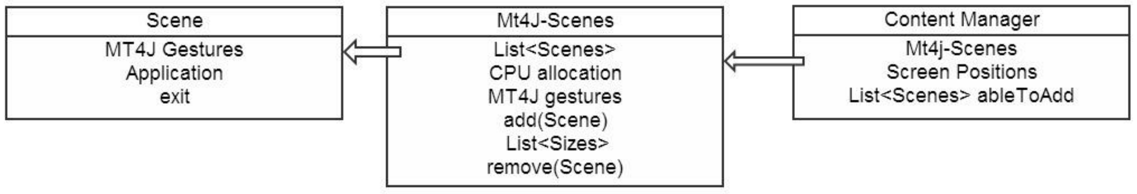
Camera Application Mockup



The camera application is
actually a combination of many
different camera novelties. It will
display a live feed to the user (using a webcam or

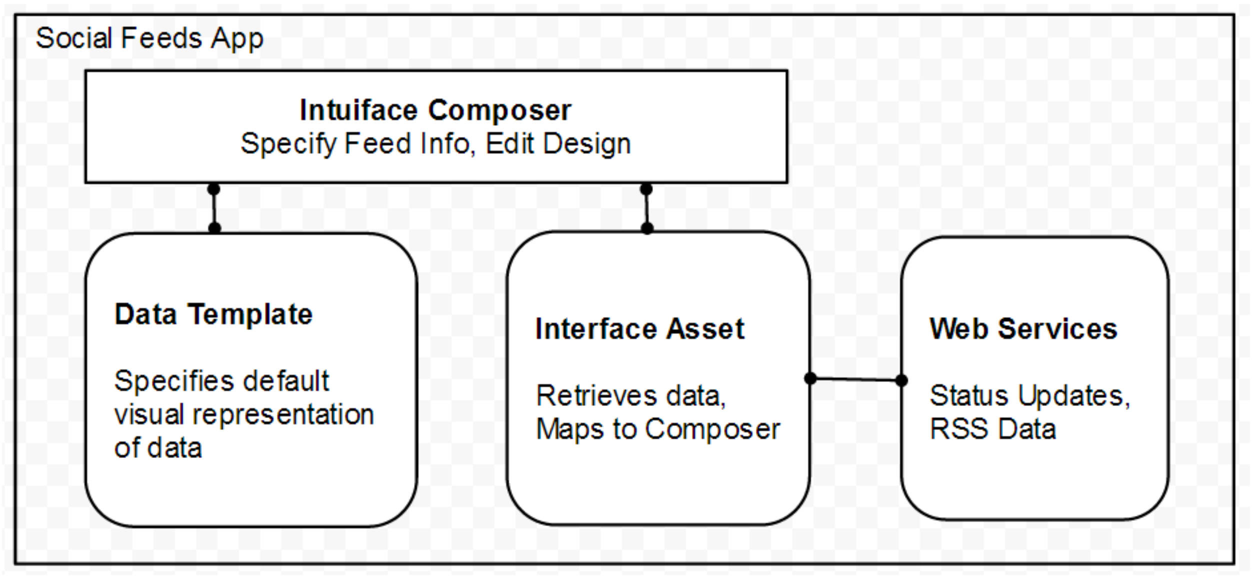
IV. Module Architecture Diagrams

Content Manager



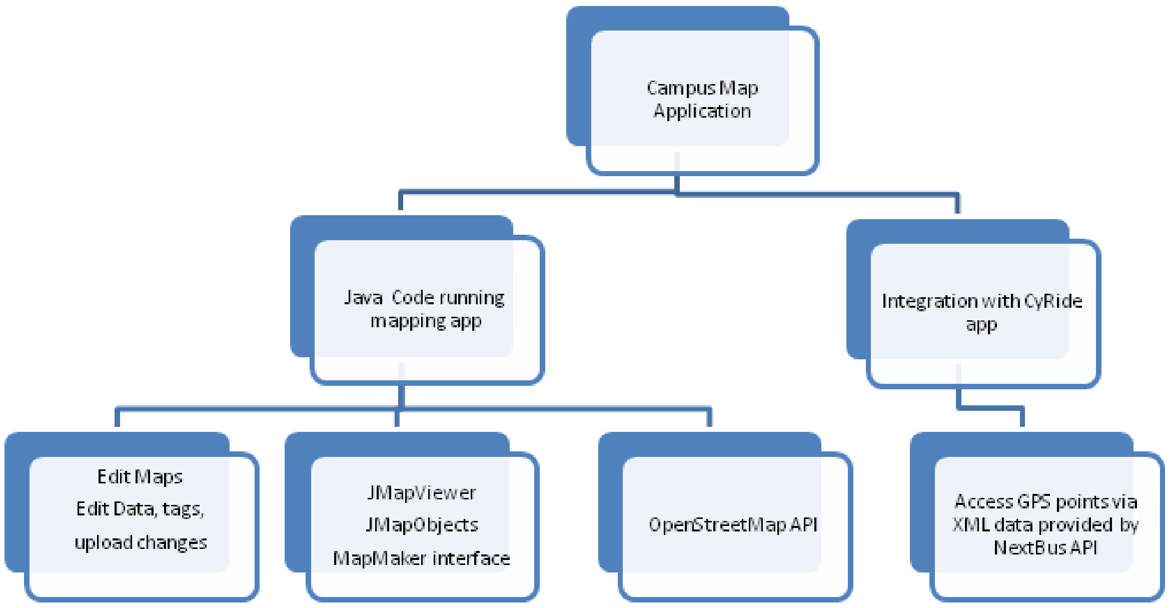
In the Content Manager Application, other programs will be launched at the touch of an icon in a scroll bar. The main requirement for this part of the project is to make it feel like a seamless integration between Intuiface and our application. The feel of the scroll bar as well as all of the gestures on the content manager need to match the Intuiface structure. MT4J handles the CPU allocation of scenes so that there can be multiple primary applications running at one time.

Feeds


Social Feeds App
Intuiface Composer
Specify Feed Info, Edit Design
Data Template
Specifies default visual representation of data
Interface Asset
Retrieves data, Maps to Composer
Web Services
Status Updates, RSS Data

The Social Feeds App will use data from web services. In Composer, the administrator will specify the feed info to retrieve, which will be retrieved by the Interface Asset via GET requests. The Data Template specifies the default visual representation of the data. Most of the complexity will hide in a custom web service we create, which will retrieve OAuth access tokens for each individual web service, handle rate limits, include necessary headers for requests, query individual web services according to specification, as well as combine, sort, and return results all from one single call from the Interface Asset.

Campus Map


Campus Map Application
Java Code running mapping app
Integration with CyRide app
Edit Maps
Edit Data, tags, upload changes
JMapViewer
JMapObjects
MapMaker interface
OpenStreetMap API
Access GPS points via XML data provided by NextBus API

In the Campus Map application, we have code running the Map through JAVA. Inside this code, we include the OpenStreetMap API, which is used to include mapping capabilities of the campus and surrounding areas. From there, we can edit the maps, the data in the maps, tags and other changes. A few classes used in the project include JMapViewer (mainly used to view the map on the display), JMapObjects (used for implementing the various objects we want to set with info about the buildings on campus), and MapMaker interface (used to actually make the maps of various places on campus).

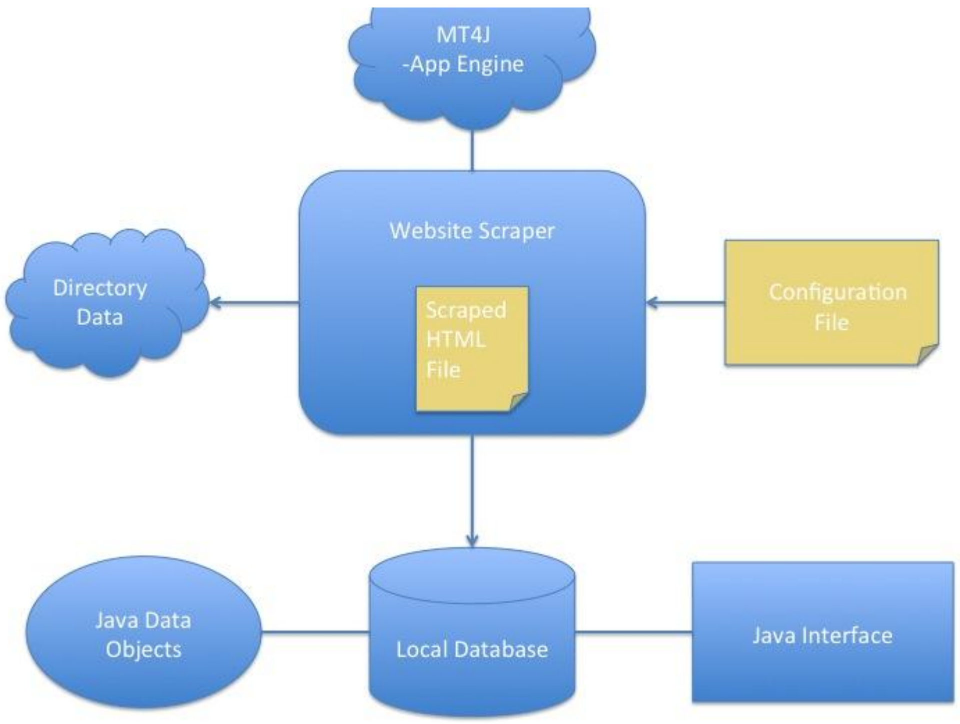
Campus Map also integrates with the CyRide application. As the CyRide app pulls in GPS data of various bus routes, these points are added into the map.

CyRide Map

The CyRide app contains a module that will periodically poll the NextBus API. When this module finds new information, the CyRide app removes old information from Campus Map and adds the new information to Campus Map.

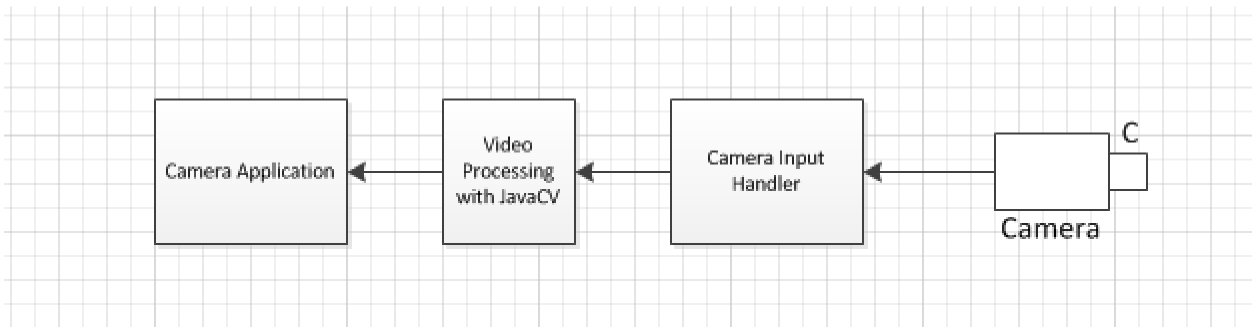
The CyRide app also contains a module that is responsible for finding a route (using the CyRide bus system) between 2 points in the city of Ames. This module then adds this route information to Campus Map and may display additional details to the user. After a timeout period, this route information is removed.

Staff Directory



In the Staff Directory Application, we will periodically parse and save data from the Iowa State University engineering staff directory into a local database. To do this we will be utilizing the java library JSoup. The database will be emptied and repopulated with each call. This data will then be displayed in a java application interfaced with the content manager. This application will display the staff directory in a similar manner to that of the online version, just without having to scroll through pages of information, and without giving the user internet access.

Camera Application



In the Camera application, we will use a stream video from a camera, using the mt4j-gstreamer-extension package. Will will attempt to first process the video using JavaCV or another video manipulation library, but it's possible that the heavy computational costs associated with encoding and decoding high resolution videos will make that infeasible.

V. System Integration

Test Cases

Content Manager Test Cases

T1: Application launching and exiting test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: The content manager should allow the user to navigate easily throughout our applications. We can assert this behavior by giving a test user a defined set of actions to perform and comparing the actual state of the system with the state we expect it should have.

T2: Simultaneous application test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: The content manager should be able to spawn a high number of child applications to accommodate periods of intense use. We can manually test this behavior by spawning multiple applications and observing the behavior of the system, checking for continued responsiveness and minimal input lag.

Feeds Test Cases

T1: Displaying feed item test

Primary Users: N/A (this test does not require a specific user)

Description: Our feed should be able to display relevant information during periods of high output and low output. We can test this by setting it to listen to a team-controlled Twitter account and mimicking periods of high activity and periods of low activity. We will then monitor what our feed displays and adjust our algorithms accordingly.

T2: Expanding feed item test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: Feed items of interest should be expandable by users who wish to learn more. This includes viewing image or other links present in the feed item. We will test to make sure users can obtain a satisfactory amount of extra information when they desire it.

Campus Map Test Cases

T1: Navigation test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: When using the Campus Map Application, the primary user should be able to navigate throughout the map of the campus without any trouble or issues. The main point we are determining from this test case is how well the map can be navigated through, and determine if there are bugs that need to be fixed from use.

T2: Sub-menu functionality test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: When inside the main map application, user has the ability of selecting a building on campus to bring us a sub-menu of options for that building. This options will include the building layout, classroom schedules, and building resources. This test will help determine how useful the sub-menus of each building is and if ordinary users can use these with ease.

CyRide Map Test Cases

T1: NextBus test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: Users should be able to expand timepoint stops on the map to see when the next bus will arrive. We can check the information seen in our application against the information displayed by the NextBus webpage on the CyRide website and make sure that it is the same.

T2: Route finding test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: The CyRide map should be able to inform users of the best bus route to take from a given point A to a given point B. We can test this by turning off all routes and enabling two at a time. This will make it clear for us as testers to see which route would be better for a user to take. We can compare the results given by our CyRide map with the results we expected.

Staff Directory Test Cases

T1: Correct information test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: The staff directory should display the data present on the online staff directory. When a user browses the staff directory, we can compare the displayed information against the information found online and check for discrepancies.

T2: Updated information test

Primary Users: Volunteers and passer-bys willing to test out the CyRIS system application

Description: When the online staff directory is updated, our staff directory application must reflect those changes. We can test this by creating a temporary REST endpoint broadcasting data that our team can change. We can set our staff directory to listen to this endpoint and make changes to the data, checking to see if they are picked up by our directory.

Camera Application Test Cases
We will not define specific test cases for the camera application at this time. This is because most functionalities we are considering simply display a feed back to the user with various transforms. Such a feature either works or doesn't, leading to marginal test cases at best.

User-Level Tests

Once we have developed a stable interface that will include applications for the CyRIS display, we will start a testing program. We will send out an email to current ISU students and staff that will used as a request to test our system. We will also ask random passer-bys that are already by the CyRIS display if they have time to help in testing our system. Our tests will be volunteer based and we will have the volunteer users go through each of the test cases and results will be recorded for future improvements.

Performance Tests

During the testing of the project, performance is a high priority. The real-time measurement of the system will be important to note. When a user is using the system and it becomes slow or loading times are not quick enough, the user will have a negative view on the project and not give the system a good review when conversing with their peers. Having a system that has minimum lag and loading times will make the interface much more user friendly and enjoyable.

There are several different tests to be used to determine the performance will of the system. The first stage of testing will be testing the speed of the system while running one application at a time. This will involve testing each of the five primary applications on its own. This will give us the performance of each application. The next stage is opening multiple applications at one time. We can then compare the speed and performance when multiple applications are opened and being used simultaneously. The final stage will be to try to overload the interface with applications, by opening multiple instances all the applications at once and determine when the system will fail to operate smoothly. When we determine the maximum number of applications that can be opened without hindering the performance of the system, we will limit the interface to this maximum number of instances at one time.

Security Tests

Testing on any system that is openly available to the public needs to go through some serious security testing. Security testing will be focused primarily on authorization, integrity, and availability (checked in the performance tests). In order for our product to become accessible on the touch wall we will need to make sure that all of these criteria have been tested.

The testing scheme for authorization will consist of users trying to use hotkeys while in programs, trying to access menus that are not available, and trying to leave the intuiface program. All of these must not happen while working with the test platform in order to move the program to the touchwall.

Integrity of the system will be tested by our different programs by running extensive tests to make sure that we are outputting true data. Since most of the data that we will be receiving is easily accessible on the web knowing that we are outputting truthful can be checked against current data.