Smart Mirror
High Risk Components

By: Suzanne Reed, Erika Zuniga, Michael Fiorenza, Tien Le
Outline

- Suzanne - APIs
- Erika - User Interaction
- Mike - Camera
- Tien - Microcomputer
High Risk Component: APIs

Suzanne Reed
Overview

The smart mirror reflects the user's image with a customizable, interactive display overlaid. The customizable display will be largely driven from the integration of APIs from various websites and companies to create “mirror widgets”. These widgets will provide both real-time data and social media interaction. Connecting to 3rd-party APIs creates considerable risk in terms of feasibility of implementation, likelihood of continued availability, and continued use costs.
Marketing Requirements

1. The system should have the ability to display widgets. Widgets should provide real-time information and allow interaction with social media sites, with a focus on photo-sharing. Priority should be given to information and social media that is likely relevant to the largest number of users.
2. The system should be relatively lower cost than current commercially available designs.
3. The system should be aesthetically pleasing.
4. The system should be easy to use and not spread germs.
### Engineering Requirements

<table>
<thead>
<tr>
<th>Marketing</th>
<th>Engineering</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>Using the combination of mirror and display, the widgets created through various APIs should appear in the user interface in an aesthetically pleasing, organized display.</td>
<td>Without widget/app functionality inlaid in the display, the device would be no different from a regular mirror. The widgets will be created using the selected APIs.</td>
</tr>
<tr>
<td>2</td>
<td>The total cost should be around $400.</td>
<td>There are smart mirrors on the market already for greater prices, the device should be affordable to consumers. To avoid contributing to costs, all APIs selected should be free.</td>
</tr>
<tr>
<td>3,4</td>
<td>The system should have touch free control.</td>
<td>Touch free control will be more aesthetically pleasing than attaching bulky peripheral inputs to the mirror. The touch free system will also be more intuitive for users than navigating menus with a mouse, and will reduce the spread of germs because there will be no direct contact with the mirror during regular use.</td>
</tr>
<tr>
<td>1</td>
<td>The system should implement API’s that can post photo to social media websites.</td>
<td>More time is spent on social media than any other major internet activity, and photo sharing is a large portion of social media activity. The mirror already uses a camera, being able to share photos will expand the user base.</td>
</tr>
</tbody>
</table>
Facebook

Good:

- Large user base
- Able to post photos
- Able to post text statuses

Bad:

- Unable to get feed stream
- Unable to get notification stream
Twitter

Good:
- Medium user base
- Able to post photos
- Able to post text tweets
- Able to get live feed

Bad:
- Functionality across several APIs
- Some functionality deprecation
Instagram

Good:
- Medium user base
- Able to pull and display photos

Bad:
- Unable to post photos
Google Feed

**Good:**
- RSA feeds have wide use
- No account or key required
- Easy implementation

**Bad:**
- Deprecated
Feedly

Good:
● RSA feeds have wide use
● Easy implementation

Bad:
● Users must create account
Good:

● No API key, no account
● Voice recognition adds input options
● Can create custom commands
● Open source

Bad:

● Small developer base
● Very specific commands
OpenWeatherMap

Good:
- Most users will want
- Lifetime guarantee of functionality for current version

Bad:
- Eventually could require paid tier
### Pugh Chart

<table>
<thead>
<tr>
<th>Importance of Functionality .25</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.6</td>
<td>.5</td>
<td>.5</td>
<td>.6</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ease of Desired Functionality .2</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>.5</td>
<td>.7</td>
<td>.1</td>
<td>.6</td>
<td>.9</td>
<td>.8</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Base Size .35</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>.8</td>
<td>.4</td>
<td>.6</td>
<td>.9</td>
<td>.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stability .2</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>.5</td>
<td>.75</td>
<td>.7</td>
<td>1</td>
<td>.2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>.73</td>
<td>.68</td>
<td>.52</td>
<td>.76</td>
<td>.66</td>
<td>.86</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>Facebook</th>
<th>Twitter</th>
<th>Instagram</th>
<th>Feedly</th>
<th>Google Feed</th>
<th>annyang</th>
<th>OpenWeatherMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Risk Mitigation

- Read and adhere to the Terms of Service
- Add redundancy to core functionality
- Only use free tier of APIs
All APIs have a free tier.
Testing

1. Make all API calls in web console, if available.
2. Make all API calls from simple web server on computer.
3. Make all API calls from the microcontroller in the intended environment.
Uncertainties

What if:

- The TOS change?
- Functionality or the API is deprecated?
- Services lose popularity?
Questions?
High Risk Component: User Interaction

Erika Zuniga
Overview

In order for the end user to interact with the Smart Mirror, there must be a web application built in the system. The UI for the system is the means for the user to get their desired information through various web services’ APIs.

A few different web application frameworks were explored in this high risk investigation.
Marketing Requirements Related to the Web Application

1. The system shall have a web application with a customizable UI.
2. The system shall have a web application with the ability to display widgets.
3. The system shall have a mobile application to interact with the full application.
4. The system shall have a database to store all user accounts.
# Engineering Specifications

<table>
<thead>
<tr>
<th>Marketing Requirement</th>
<th>Engineering Specification</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Web application that allows for a customizable interface shall be hosted on the chosen microcomputer.</td>
<td>A web application is necessary to display the content requested by the user. It will be customizable by the user since not all users will have the same preferences.</td>
</tr>
<tr>
<td>2</td>
<td>Widgets from various web services will be part of the web application.</td>
<td>The Smart Mirror would only function as a regular mirror if there were no widgets or applications to interact with.</td>
</tr>
<tr>
<td>3</td>
<td>Mobile application shall be linked to the full web application.</td>
<td>To begin using the Smart Mirror, it is necessary to create an account via mobile application associated with the main application because the user must upload sample photos for facial recognition in the login process. Through this, the user may choose some APIs they want to interact with.</td>
</tr>
<tr>
<td>4</td>
<td>A database shall be used to store all users’ accounts and preferences.</td>
<td>All of the information pertaining to all of the users’ accounts and preferences must be kept in a data storage mechanism.</td>
</tr>
</tbody>
</table>
Risk Investigation

- **Web application frameworks considered for the Smart Mirror system:**
  - Ruby on Rails
  - Django
  - Tapestry5

- **Layout of the Web Application**
  - Web Application Layers
  - Server Client Model
Ruby on Rails

- Popular choice in web application development
- Uses MVC design pattern
- Convention over Configuration (structured layout) is not ideal
- Team members do not have much experience programming in Ruby
Django (Python)

- Another popular choice in web app development
- Python is easy to use (All members have experience)
- Large user base, documentation
- “MTV” framework (similar to MVC)
- Developer has more control in choosing the layout and configuration (necessary for the Smart Mirror system)
Tapestry5 (Java)

- Another framework with a programming language the team has experience with
- Smaller user base
- Designed to operate on JVM’s
## Pugh Analysis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Django</th>
<th>Ruby on Rails</th>
<th>Apache Tapestry5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Software License</td>
<td>3</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>User base</td>
<td>4</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>Customizable</td>
<td>4</td>
<td>+1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Testing Frameworks</td>
<td>3</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>Security Frameworks</td>
<td>3</td>
<td>+1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17</strong></td>
<td><strong>10</strong></td>
<td><strong>6</strong></td>
<td></td>
</tr>
<tr>
<td>Consider?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Risk Mitigation

- Ultimately, the Django Framework with Python was chosen.
- The mobile component to this system will be a client of the main application.
Risk Mitigation (cont’d)

- MySQL for the database server (can support Django)
- UI can be customized with Bootstrap (front-end framework)
- Apache for deployment
Testing

- Unit Testing for each component of the system
- Mobile Client: Registration, uploading photos, connecting to facial recognition software - overall interaction with the main application
- Main application: UI navigation
Uncertainties

- Although Python is easy to use, getting familiar with the tools provided by the Django framework may take some time (developer entry barriers).
- The chosen microcomputer to host the web application will need high processing power to keep up with all the user interactions.
Questions?
High Risk Component: Camera

Michael Fiorenza
Overview

● A camera needed to be selected to allow the user to interface with the Smart Mirror through facial recognition, as well as, allow for navigation.

● Navigation of the user interface will allow for body gestures to complete different tasks.

● When selecting a camera, the following areas were considered to be the most important:
  ○ Cost
  ○ Usability (Open Source Libraries)
  ○ Capability to obtain desired functionality
Risk Specification

Marketing Requirements related to the camera:

1. The system should have the ability to detect the user.
2. The system should have the ability to navigate the UI based on user motion.
3. The system should have a relatively lower cost than implemented designs.
<table>
<thead>
<tr>
<th>Engineering Requirement</th>
<th>Marketing Requirement Number</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system should accurately perform facial recognition to distinguish users.</td>
<td>1</td>
<td>This allows for multiple users to control the system and allow for a customizable UI.</td>
</tr>
<tr>
<td>The system should be controllable by user motion without the need for a mouse or keyboard.</td>
<td>2</td>
<td>The only time a keyboard should be attached to the system is when maintenance is required. The added peripherals will not look aesthetically appealing to the overall design.</td>
</tr>
<tr>
<td>The total cost should be around the group budget of $400</td>
<td>3</td>
<td>There are smart mirrors in the market already for greater prices, the device should be affordable for consumers.</td>
</tr>
</tbody>
</table>
Three different cameras were investigated in depth:

- Microsoft Kinect
- Asus Xtion Pro Live
- Webcam
Microsoft Kinect  Asus Xtion Pro  Webcam
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Microsoft Kinect</th>
<th>Webcam</th>
<th>Asus Xtion Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>.25</td>
<td>0.6</td>
<td>0.9</td>
<td>0.3</td>
</tr>
<tr>
<td>Desired Functionality</td>
<td>.5</td>
<td>0.95</td>
<td>0.15</td>
<td>0.95</td>
</tr>
<tr>
<td>Community Support</td>
<td>.25</td>
<td>0.9</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>0.85</td>
<td>0.475</td>
<td>0.675</td>
</tr>
<tr>
<td>Consider Using?</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Pugh Table Analysis of Cameras
Microsoft Kinect

- Large community support and Open Source Libraries for Linux
- Pre-written examples for capturing body gestures
Risk Mitigation Design

- The Microsoft Kinect uses an RGB camera with a depth sensor and infrared projector with a CMOS sensor which is able to generate a 3D environment.
- The captured images are sent over USB 2.0 which requires about 50% bandwidth (20-30MB/s).
- Need a microcontroller capable of processing the images, as well as, allow for enough throughput over USB 2.0
<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Market Price</th>
<th>Actual Paid Price</th>
<th>Availability: Website Shipping Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Kinect</td>
<td>RGB Camera with depth sensor capable of generating 3D images</td>
<td>$165</td>
<td>$0 (owned)</td>
<td><a href="http://www.amazon.com/Kinect-Sensor-Adventures-Xbox-360/dp/B002BSA298/ref=sr_1_fkmr0_1?ie=UTF8&amp;qid=1447699057&amp;sr=8-1-fkmr0&amp;keywords=microsoft+kinect+v1">link</a> (Amazon Prime)</td>
</tr>
</tbody>
</table>
Testing Strategy

- Verify that the Kinect can produce RGB and Depth images at 30fps and send these images over USB 2.0 and ensure there is no frame loss
- Verify that the microcontroller can process these images to perform facial recognition and gesture recognition
- These two verifications will ensure that the microcontroller and camera are compatible with meeting the Engineering Requirements
Uncertainties

- Finding a microcontroller that is capable of processing the images
  - Raspberry PI 2 lacked the required USB 2.0 throughput
- Accurately performing facial recognition
  - OpenCV provides libraries and documentation
- Accurately recognizing body gestures
  - PrimeSense drivers provide libraries and documentation
Questions?
High Risk Component: Microcomputer

Tien Le
Overview

- The Microcomputer choice directly impacts the overall effectiveness of the entire device.
- Using a low-profile computer would be too costly and would not look very good attached to the mirror.
- Using an old laptop would reduce the price of constructing the mirror but not make it easily reproducible.
- Interfacing through a camera requires a large amount of image processing, so a suitable microcomputer which can support that without being too large or expensive is necessary.
- Alongside interfacing with the Kinect, the microcomputer needs to be capable of displaying all of the information.
- Five different microcomputers were researched and examined in depth.
Risk Specification

In order to design this mirror, marketing requirements were established. Out of the total collection of marketing requirements, the following subset is relevant to this component selection:

1. The system should have the ability to detect the presence of the user.
2. The system should have the ability to navigate the UI based on user motion.
3. The system should have a relatively lower cost than existing designs.
<table>
<thead>
<tr>
<th>Engineering Requirement</th>
<th>Marketing Requirement Number</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system should accurately perform facial recognition within 20 seconds to distinguish between different users.</td>
<td>1</td>
<td>This allows for multiple users to control the system and allow for a customizable UI. If the microcomputer does not have enough computing power to perform facial detection, the mirror would not be very usable.</td>
</tr>
<tr>
<td>The system should be controlled by user motions without the need for a mouse or keyboard.</td>
<td>2</td>
<td>The only time a keyboard should be attached is when maintenance is required. The added peripherals will not look aesthetically appealing to the overall design.</td>
</tr>
<tr>
<td>The total cost should be around $400</td>
<td>3</td>
<td>There are smart mirrors in the market already for greater prices, the device should be affordable for consumers.</td>
</tr>
</tbody>
</table>
Risk Investigation

Five different Microcomputers were investigated in depth:

- Raspberry Pi II
- Banana Pi
- BeagleBone Black
- ODROID C1
- ODROID XU4
Raspberry Pi II

- Everyone knows about these
- Developed by the Raspberry Pi foundation
- Can be found for around $35
- 900 MHz Quad-core ARM cortex A7 CPU
- VideoCore IV dual-core GPU
- 1 GB of DDR2 RAM
- Supports various Linux Distros + Windows 10 IoT core
Banana Pi

- Developed by Sinovoip, a Chinese company
- Also found for $35
- There are higher-spec models but they are hard to get
- 1GHz dual core Cortex A7 CPU
- Mali 400 MP2 dual core GPU
- 1 GB DDR3 RAM
- Android + Various Linux Distros
BeagleBone Black

- Produced by Texas Instruments
- Found for $55
- 1GHz ARM Cortex A8 CPU
- PowerVR SGX530 GPU
- 512MB DDR3 RAM
- Android + Various Linux Distros
ODROID C1

- Developed by Hardkernel, a South Korean company
- Also found for $35
- 1.5GHz quad-core Cortex A5 CPU
- Mali-450 Mp2 GPU
- 1GB DDR3 RAM
- Android + Various Linux Distros
ODROID XU4

- Developed by Hardkernel, a South Korean company
- Found for $75
- 2 GHz Octa-core Cortex A5 CPU
- Mali-T628 M6 GPU
- 2GB DDR3 RAM
- Android + Various Linux Distros
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Raspberry Pi II</th>
<th>Banana Pi</th>
<th>BeagleBone Black</th>
<th>ODROID C1</th>
<th>ODROID XU4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphics Processing</strong></td>
<td>5</td>
<td>Baseline</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>3</td>
<td>Baseline</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td><strong>Ease of development</strong></td>
<td>2</td>
<td>Baseline</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Computational Power</strong></td>
<td>4</td>
<td>Baseline</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td><strong>Development Popularity</strong></td>
<td>1</td>
<td>Baseline</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>Baseline</td>
<td>-10</td>
<td>-3</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Consider Using?</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Pugh Table Analysis of Microcomputers
(Raspberry Pi II as Baseline)
Final Verdict?
• When a Raspberry Pi was initially connected to the Kinect, performance was not up to par.
• Despite increased cost, the greater computing capabilities are worth it.
• The Kinect is very resource intensive and in order to have a smooth user experience, a powerful computer is required.
• Cheaper than building a low-profile PC.
Risk Mitigation Design

- Available budget is relatively low when considering the cost of the mirror display itself.
- Having a powerful microcomputer would still be cheaper than creating a low-profile computer or using an old laptop.
- The low performance of the Kinect with the Raspberry Pi II demonstrated that the UI would not be very receptive to user motion.
- In order to have fast facial recognition with the Kinect (1st engineering requirement), a powerful computer is required.
Risk Mitigation Design (Cont)

- There is a minimum number of USB ports required in order to get the intended functionality.
- One problem with the Raspberry Pi is the fact that the Network Interface Controller and the USB Ports all share the same interface to save space.
- XU4 has USB 3.0 and will not have the same bottlenecks.

Diagram of Smart Mirror
<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Market Price</th>
<th>Actual Paid Price</th>
<th>Availability: Website + Shipping Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODroid XU4</td>
<td>The microcomputer</td>
<td>$75</td>
<td>$0</td>
<td><a href="http://ameridroid.com/products/odroid-xu4">http://ameridroid.com/products/odroid-xu4</a> 1-2 Weeks Approx.</td>
</tr>
<tr>
<td>MicroUSB charger</td>
<td>MicroUSB cable used to power device</td>
<td>$5</td>
<td>$0</td>
<td><a href="http://www.amazon.com/">http://www.amazon.com/</a> (Prime) 2 Days</td>
</tr>
<tr>
<td>HDMI Cable</td>
<td>Required for video output</td>
<td>$5.49</td>
<td>$0</td>
<td><a href="http://www.amazon.com/">http://www.amazon.com/</a> (Prime) 2 Days</td>
</tr>
<tr>
<td>microSD Card 8 GB</td>
<td>Used to store operating system</td>
<td>$6.30</td>
<td>$0</td>
<td><a href="http://www.amazon.com/">http://www.amazon.com/</a> (Prime) 2 Days</td>
</tr>
</tbody>
</table>
Testing Strategy

- There are multiple benchmarks that can be run on Ubuntu that can determine the overall computing capabilities of the device.
  - Processor speed tests
  - Heavy load RAM stress tests
  - Graphical Benchmarks

- Verify that fast Kinect functionality is achievable with the device.
  - ~20-30 FPS seen on camera output
  - Confirm that the motion detection works well enough to be used in a user interface.
Uncertainties

- After initial tests with the device, it appears to run a bit hot.
  - Long term deployment of the device hasn’t been tested yet and cannot really be tested until the amount of computing work that the device will perform is known.
- None of the microcomputers are waterproof/water resistant.
  - If the Smart Mirror is left in a bathroom how will everyday usage affect the functionality of the microcomputer?
- None of this can really be tested until a working prototype is made and possibly field testing is performed.
Questions?