Service Blueprints, Wire Frames, and Use Cases

First, we take a ridersharing scenario and cast it as a service blueprint.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Physical Evidence** | **Customer Actions** | **Onstage/Client Phone** | **Backstage** | **Support/Server** |
| App on iPhone | Phil decides to offer rides and uses his phone to start the Ride-share app , selecting Offer Ride | Offer Ride screen appears. | App queries Phil’s current location and other information to initialize screen. | System notes Phil’s activity and begins active monitoring. |
|  | Phil fills out needed information about his planned trip. |  | Check and send to server. | Put Phil’s offered trip(s) into routing data base. |
|  | Harry decides he is ready within a 20 minute window to go to work |  |  |  |
| App on iPhone | Harry takes out his iPhone, selects the RideShare App and goes to Request Ride screen, fills out details of desired ride, offering $3. | After selection is made, shows a “get back to you soon” message | App queries Harry’s current location. App sends request along with Harry’s current location to server. | System receives request. Starts to monitor Harry. Searches for suitable matches. Prioritizes current location, destination, friend match via Facebook, financial compatibity, reliability and congeniality ratings. It selects Phil, and sends him a request. |
| Phil’s phone rings, |  | Shows Harry’s ride request, deducting $0.50 from Harry’s offer. |  |  |
|  | Phil looks at request and says “Yes” confirming contact information. | “Thank you, will connect you..” Transitions to a map view showing Harry’s location and an estimate of arrival time. | Sends information to server, and begins to monitor Phil’s location frequently. | Creates *trip contract* and sends confirmation to Harry. |
| Harry’s phone rings. | Harry sees ride is on and sends SMS to Phil saying “Thanks, I’ll be there.” |  |  | System recalculates estimated arrival time. Sends location and estimate to both phones every 60 seconds |
|  | Harry and Phil monitor progress sproadiaclly. | App updates map view and arrival estimate based on Phil’s location. |  |  |
| Phil’s car shows up outside of Harry’s home | Harry exits his home and climbs into Phil’s car | Both phone show ride as “in progress”. | Phil and Harry’s phones connect via Bluetooth. Send message to server that they are co-located followed by periodic location updates. | System changes ride status from tentative to active. Monitors progress. |
| Harry is inside of Phil’s car |  | Both phone shows current location and estimated arrival time |  |  |
| Arrival in front of Harry’s office | Harry exits Phil’s car | App update to show that the trip is complete. | Phone detects arrival location, detects break in Bluetooth connection. Sends message to server that ride is complete | Server sends feedback requests to both Phil and Harry. |
| Both phones ring |  | Feedback screen appears on phones |  |  |
|  | Harry confirms successful trip. | “Thank you.” | Sends confirm. | Move $2.50 from Harry’s acount to Phil’s. Send notification to Phil. Update Phil’s rating. |
| Phil arrives at his destination. | Responds to feedback with “Trip was bad. Harry is a jerk.” | “Thank you. Sorry about that.” | Send feedback to server. | Server receives response and updates Harry’s ratings and puts him on Phil’s *black list*. |

Use cases are supposed to describe a short, continuous encounter of the user with the system, so this scenario gets broken up into multiple use cases. Once you have made a wire frame, a lot of repetition can be eliminated from the use case by referring to the information displayed and entered on the wireframe. It is also good to carry the Onstage, Backstage, Support distinctions you have already made into the use cases.

We’ve boiled all these modifications down into the following templates.

(This was copied from PowerPoint)

### A \* indicates that an item is optional. A preset suggests the field may be initialized with a default value. Gray indicates something else must happen before the item is ready to be clicked.

### Use Case

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| **Preconditions** | | | | |
|  |  |  |  |  |
| **Guarantees** | | | | |
|  |  |  |  |  |
| **Main Success Scenario** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Extensions** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |

**The Ridesharing Use Cases**

Here is a comprehensive list of use cases needed to specify the whole system. We will only do a few of them.They may be a little more elaborate than necessary.

Subscriber (either driver or rider):

1. Register
2. Log in
3. Manage Profile
4. Billing
5. Confirm Trip

Driver:

1. Offer Ride
2. Accept a rider
3. Cancel Drive
4. Take Trip

Rider:

1. Request Ride
2. Accept driver
3. Cancel Ride
4. Join Trip

Information System:

1. Monitor Trips
2. Schedule Repeated Offers and Requests

Operator:

1. Handle Emergency
2. Follow-up to resolve problems or gather feedback.
3. Analyze and Manage Performance

### This one is a computer screen, not a phone.Manage Profile

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User: Subscriber** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
| Get set up so I can arrange rides with others |  |  |  |  |
| Control preferences |  |  |  |  |
| **Preconditions** | | | | |
|  |  |  |  |  |
| **Guarantees** | | | | |
|  |  |  |  | Success: Profile stored with all needed information. |
|  |  |  |  | Minimal: No information lost or corrupted. |
| **Main Success Scenario** | | | | |
|  | 1. Change Name, Password, B’day. |  | Store |  |
|  | 2. Change email. | Please respond to the email we’re sending. | Remember pending flag | Send query to email |
|  | 3. Change phone | Please respond to the SMS message we’re sending with “Go”. | Remember pending flag | Send SMS to phone, |
|  | 4. Submit | Return to previous page | If consistent, complete, and checked, send to server. | Store. |
|  | 5. Set Billing Information | Go to Billing |  |  |
|  | 6. |  |  | When receiving response from phone or email,  Remove pending flag |
| **Extensions** | | | | |
|  | 4a. Fail to fill in mandatory field | Flag unfilled field and stay on view. |  |  |
|  | 4b. | Repeat appropriate “Please respond” message, including resend option. | Pending flag not removed. |  |
|  | 4c. Neither driving nor riding selected | Complain and stay on view. |  |  |

### Offer Rides

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User: Subscriber** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
| Wants to save money/time. |  |  |  | Wants commissions |
| Social Networking |  |  |  |  |
| Environmentalism |  |  |  |  |
| **Preconditions** | | | | |
|  | Driver has registered |  |  | Profile is set up properly for a driver. |
| **Guarantees** | | | | |
|  |  |  |  | Success: Trips stored in data base |
|  |  |  |  | Minimal: Problem stored in analytics data. |
| **Main Success Scenario** | | | | |
|  | 1. Fill in return time |  | Create and store second trip values. |  |
|  | 2. Fills in other text fields or option |  | Store |  |
|  | 3. Make Offer |  | Check for completeness, consistency. If Repeat box checked, store trip in Repeating Schedule. Send to sever. | Call Google Directions to plot route unless she specified route. Store trip(s) in routing data base. |
| **Extensions** | | | | |
|  | 4. Specify Route | Overlay Google Maps route, allow fiddling, followed by “OK” | Change route plan. | Communicate between Google Maps and phone. |
|  | 3a. Mandatory field missing. | Flag field and stay on view. |  |  |
|  | 3b. | Asks for clarification of place. |  | If either place cannot be found in map database |
|  | 3c. Return time is before departure time | Complain and stay |  |  |

### Request a Ride

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User: Rider** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
| Rider wants limited delay, reasonable cost |  |  |  | Wants the commissions |
| Request a ride so that she can avoid driving and be safe |  |  |  |  |
| **Preconditions** | | | | |
|  | App running |  |  | Rider has an account, complete data |
| **Guarantees** | | | | |
|  |  |  |  | Success: Trips stored in data base |
|  |  |  |  | Failure: Problem stored in analytics data. |
| **Main Success Scenario** | | | | |
|  | 1. Fills in Date, Start time, Start place, Offer |  | Store |  |
|  | 2. Fill in return time |  | Create and store second trip values. |  |
|  | 3. Fill in end place | Calculate Offer from profile if blank. |  |  |
|  | 4. Request | Return to previous view. | Check for completeness, consistency. Send to server. | Search Routing data base for best driver. If Repeat box checked, store trips in Repeating Schedule. |
| **Extensions** | | | | |
|  | 4a. Mandatory field missing. | Flag field and stay on view. |  |  |
|  | 4b. | Asks for clarification of place if… |  | … either place cannot be found in map database |
|  | 4c. Return time is before departure time | Complain and stay |  |  |
|  | 4d. | Complain and overlay payment window if… | … offer exceeds balance |  |

### 

### Accept a Rider

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User: Driver** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
| See Offer a Ride |  |  |  | Wants to make matches |
| **Preconditions** | | | | |
|  |  |  |  | A feasible driver/rider match has been found, see Request a Ride. |
| **Guarantees** | | | | |
|  |  |  |  | Success: Trip contract is stored in both personal databases. Trip scheduled  Failure: Route database stays the same. |
| **Main Success Scenario** | | | | |
|  | 1 | Request is sent to driver’s real-time contact device(s) initialized with driver’s default box checks. |  |  |
| Phone rings. I know/don’t know this person | 2. Change check boxes or message |  | Store |  |
| Looks good. | 3. Yes | Thank you. close |  | Store trip. Notify rider on contact device(s) including details. Leave driver in route database unless capacity reached. |
|  |  |  |  |  |
| **Extensions** | | | | |
| I have questions | 4. Maybe |  | Send Message, contact information to rider |  |
|  | 5. No | Thank you. close | If Message, Send to rider. |  |
| Driver may contact rider by phone or email. |  |  |  |  |

### Confirm Ride

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User: Driver or Rider** | | **System** | | |
| **Physical Evidence** | **Actions** | **Onstage** | **Backstage** | **Support** |
| **Goals and Motivations** | | | | |
| Knows what happened recently. |  |  |  |  |
|  |  |  |  |  |
| **Preconditions** | | | | |
| Trip has ended, or should have. |  |  |  |  |
| **Guarantees** | | | | |
|  |  |  |  | Success: All participants respond and results are stored. |
|  |  |  |  | Failure: Case is sent to review process, no change to data base |
| **Main Success Scenario** | | | | |
| Phone rings. | 1. Opens app | Confirmation requested. |  |  |
|  | 2. Fills in details |  | Store for sending later. |  |
| Happy | 3.Rider checks good and Send | Thank you |  | Money is transferred from rider’s account to driver’s. |
| Happy | 4. Driver checks good and Send. | Thank you. |  | If Rider has not responded, request again or send to urgent follow-up process. |
| Something went very wrong. | 5. Rider or driver checks aborted and Send. | Sorry. Close |  | Follow-up process is initiated. No money transferred. |
| Something went wrong. | Rider or driver checks bad and Send. | Sorry, Close. |  | Follow-up process is initiated. No money transferred. |
| **Extensions** | | | | |