Overview (Diana)

Description

The Wheel Deal provides a platform by which people can gain access to bikes by renting them from bike owners, in exchange for some small rental fee. Renters can search for bikes using filters such as bike type, date and time, location, and price. Before reserving a bike, users must create accounts using a valid phone number, and provide payment information. Bike owners provide specs, time availability, location, and price for each of their bike listings. Owners and renters have the ability to leave reviews about the experience.

Purpose

Our first and foremost priority at the Wheel Deal is to get people to where they want to go. We'd like to do more than just getting people from place A to place B; we want to use our platform to move the world towards a greener, healthier, and more sustainable future. We believe that we can accomplish this in two ways: by making bikes more accessible regardless of geographical location, and by reducing the amount of waste caused by discarded (but often still functional) bikes.

The first step in achieving our two-part goal is to increase overall bike accessibility. Only 46% of people in the US have access to a bike, and the main reason why people choose not to bike is because they don't have access to one. There are bike-sharing companies that allow you to rent bikes, but most of these services only exist in major cities, so all of the people outside these cities who do not own a bike are out of luck. Oftentimes, people who don't have access to bikes rely on cars instead, increasing the amount of greenhouse gas emissions in the atmosphere. In 2016, cars were the source of 60% of CO2 emissions by modes of transport in the EU. By getting more people out of their cars and onto bikes, we can cut this number down significantly. In Philadelphia alone, bikers ride 260,000 miles daily, preventing 47,450 tons of carbon dioxide emissions per year. And, people who choose to bike have also gained health benefits. From increasing bike usage from 4 to 24 minutes, riders have seen a 14% reduction in cardiovascular disease and diabetes. By increasing overall bike accessibility and getting more people on bikes, we can improve the health of the environment and the health of the people.

The second step in achieving our two-part goal is to reduce bike waste. Many bike-sharing services today make and use their own custom bikes. While this lends itself to a pretty fleet of pristine and uniform-looking bikes, after all of these bikes have seen wear and tear, where do they go? Many of them end up in large, mountainous "bicycle graveyards." An additional 15 million bikes end up discarded by their owners every year, many of which still function fine, because many people who own bikes end up rarely using them. We want to focus on getting more people to use pre-existing bikes, thereby reducing the waste caused by making and discarding bicycles.

Existing Solutions

Currently, there are three main systems of bike rental: docked bikes, dockless bikes, and touring bikes. Docked bike systems like BlueBikes set up central docking stations throughout the city, allowing users to pick up a bike at one docking station and return them to another. Dockless bike systems like Lime or Jump forego the use of centralized docking areas entirely, and users can pick up and leave bikes anywhere within the city. Touring bikes specifically target tourists who want to explore the city by bike, with one central pick-up and drop-off location. Between all of these bikesharing services, it's become easier to locate nearby bikes and cheaper to travel from one location to another. Many major cities have also seen significant environmental and public health benefits. However, each individual service comes with its own set of drawbacks.

Docked systems like BlueBikes are typically set up in a large scale by a single company, with fixed stations throughout the city. While having docking stations can make finding bikes easier, the fixed stations may also add a layer of inconvenience depending on where the user is biking to and/or from. When a user is in need of a bike, there is no guarantee that a bike will be available for use at their closest docking station. Conversely, when a user would like to return a bike, the closest docking station may be full and the user may need to go out of their way to find an open dock, spending extra time and, potentially, money.

Dockless systems forego the use of docking systems completely, thereby avoiding many of the problems associated with docked systems. However, there is no guarantee that a bike will be available when needed and, more pressingly, leaving bikes throughout the city can mean people leave them in inappropriate or even dangerous places.

Most major cities also offer rental bikes, targeting tourists who want to explore the city on a bike. However, companies offering these bikes typically only have one central location, so riders often have to go out of their way to pick up and drop off bikes. Furthermore, these bikes can get expensive and, depending on the season, riders may even have to swim through crowds of tourists to rent a bike.

Finally, all of the above systems share the problem that they are mainly offered in major cities. That means if someone didn't live in a major city with these services, or lived outside the city boundaries of where these services are offered, they most likely would not have access to a bike rental service.

https://www.europarl.europa.eu/news/en/headlines/society/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics

http://peopleforbikes.org/our-work/statistics/statistics-category/?cat=environmental-statistics

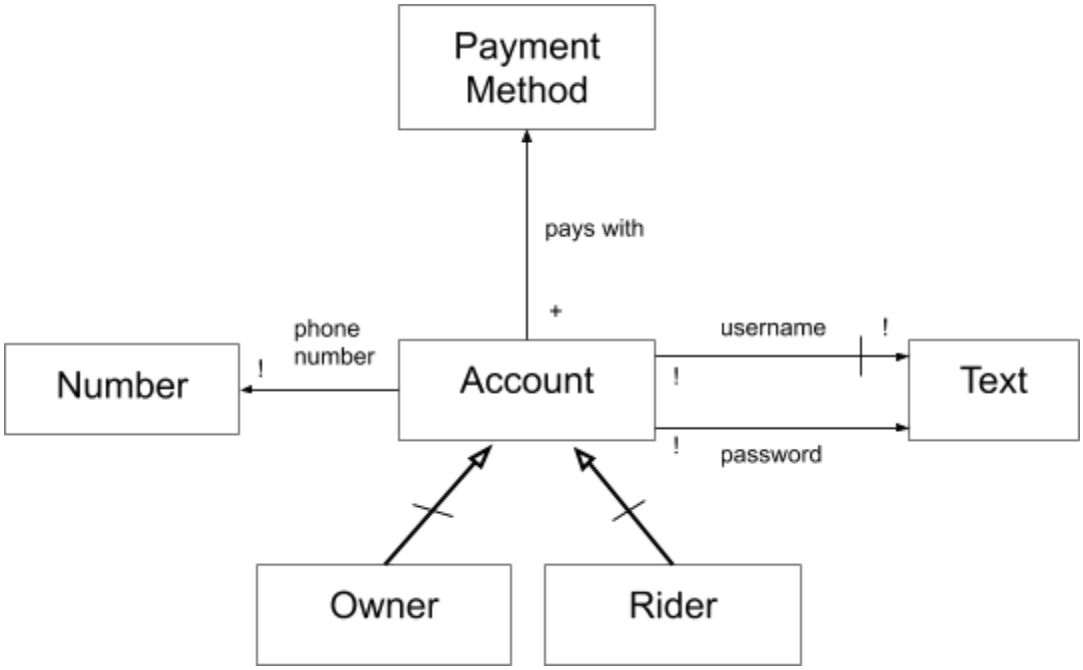
https://www.washingtonpost.com/news/dr-gridlock/wp/2017/10/05/abandoned-vandalized-and-illegally-parked-bike-share-bikes-now-a-d-c-problem/

Conceptual Design (Jinny)

Account

Purpose: ensure credibility of users

Structure:



Actions:

- CreateAccount(un: Text, pw: Text, pn: Number): Account
  no username.un =>
  username += un; password += un -> pw; phone_number += un -> pn;
- AddPaymentMethod(pm: PaymentMethod)
  username.un =>
  pays_with += un -> pm;

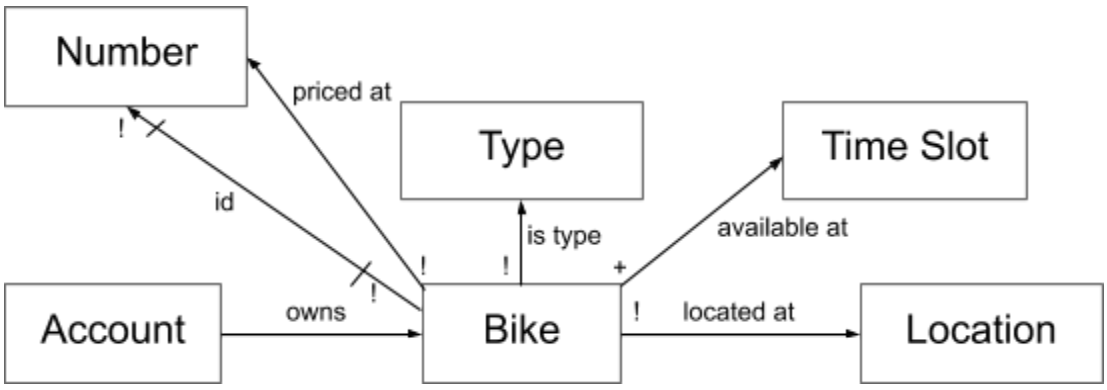
Tactics:

- If CreateAccount is successful, AddPaymentMethod
- After CreateAccount and finishing their ride, the user decides to AddUserReview for the owner of the bike

Bike

Purpose: exchange reliable method of transit for currency

Structure:



Actions:

- CreateBike(a: Account, t: Type, bike_id: Number): Bike
  no id.bike_id =>
    owns += a -> bike_id; is_type += bike_id -> t;
- DeleteBike(bike_id: Number, owner: Account)
  id.bike_id =>
    id -= bike_id;
    priced_at -= bike_id;
    is_type -= bike_id;
    available_at -= bike_id;
    located_at -= bike_id;
    has_reviews -= bike_id;
    owns -= owner -> bike_id;
- SetLocation(l: Location, bike_id: Number)
  id.bike_id =>
    located_at += bike_id -> l;
- SetAvailability(availability: TimeSlot, bike_id: Number)
  id.bike_id =>
    available_at += bike_id -> availability;
- SetPrice(p: number, bike_id: Number)
  id.bike_id =>
    priced_at += bike_id -> p;
- SeeBikes(l: Location): set Bike - see available bikes based on location
  result = { b | b -> l in located_at };

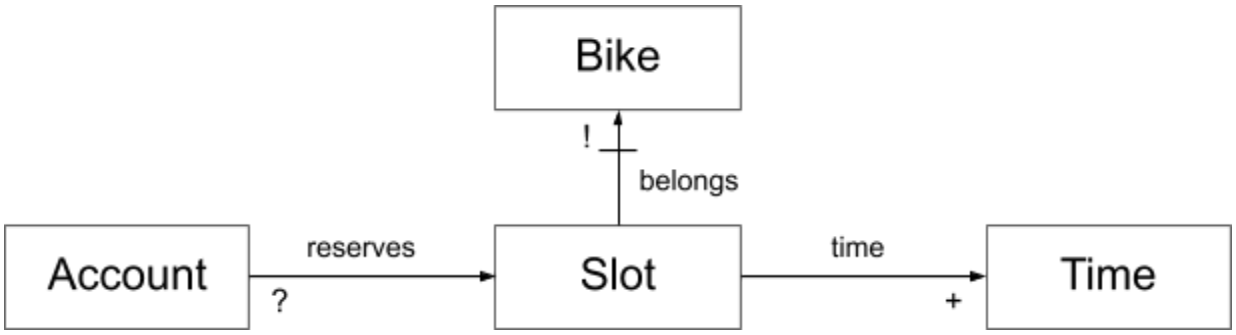
Tactics:

- Owner:
    - If CreateBike is successful, the owner SetPrice, SetLocation, and SetAvailability of the bike to start lending bike out to other users
    - If the owner wants to update their bike information, they can once again SetPrice, SetLocation, and/or SetAvailability to meet their circumstances
    - If bike becomes permanently no longer available, DeleteBike
- Rider:
    - While the rider is looking for a bike, they can SeeBikes nearest to their desired location; after the rider has completed their ride, they may decide to AddBikeReview for others' reference

Ride

Purpose: guarantee a bike for a specific time frame and location

Structure:



Actions:

- CreateRide(b: Bike, s: Slot): Ride
  no belongs.s =>
  belongs += s -> b;
- ReserveRide(a: Account, s: Slot, b: Bike, t: Time)
  s -> b in belongs =>
  reserves += a -> s;
  time += s -> t
- CancelRide(a: Account, s: Slot)
  a -> s in reserves =>
  reserves -= a -> s;
- UseRide(a: Account, s: Slot, b: Bike)
  a -> s in reserves and b -> s in belongs =>
  reserves -= a -> s;

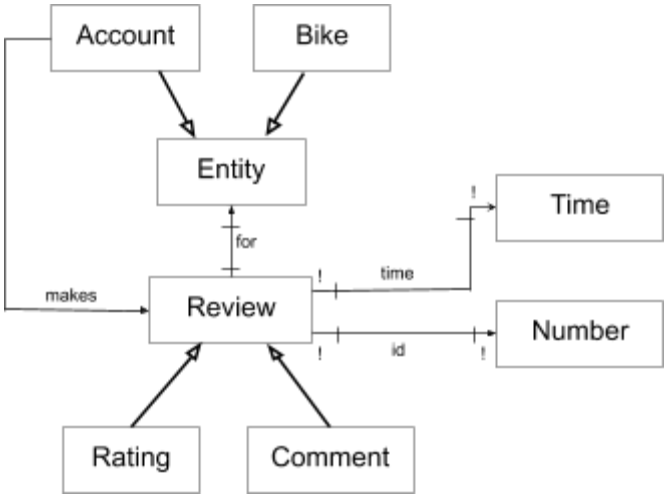
Tactics:

- If CreateRide is successful, and ReserveRide and no CancelRide, then can UseRide and pick up the given bike at the given time slot and location

Review

Purpose: provide useful feedback about an entity for other users

Structure:



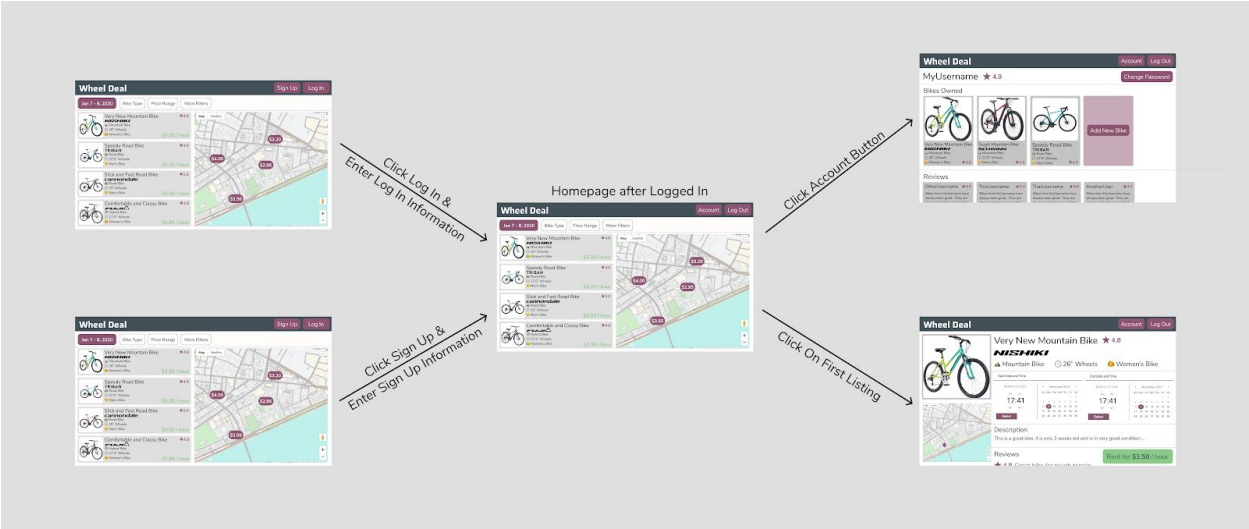
Actions:

- AddReview(r: Rating, c: Comment, a: Account, e: Entity, t: Time, review_id: Number): Review
  no id.review_id =>
  id += review_id -> { r, c, t };
  makes += a -> review_id;
  for += review_id -> e;
- EditReview(review_id: Number, new_r: Rating, new_c: Comment)
  id.review_id =>
  id.review_id.r = new_r;
  id.review_id.c = new_c;
- ComputeOverallRating(e: Entity): Number
  result = { id.review_id.r * weight(id.review_id.r) | review_id -> e in for };
  weight is derived based on time the review was created

Tactics:

- After finishing their ride, the user decides to AddReview for the bike they rented, and AddReview for the bike owner's help with answering questions about the bike
- The user realized they miswrote something in their previous review, so they decide to EditReview

Wireframes (Linnea)



https://www.figma.com/file/RoH856bq42b4hXpkMY0JWc/Untitled?node-id=0%3A1

Design Commentary (Linnea)

Who can write reviews for bikes and users? How many reviews can be written? Can reviews be edited?

While allowing anyone to reviews for every bike and every user would be much easier to implement, it is important to limit the reviews to people who have actually used a bike or interacted with an owner. Thus, we decided that user X may only make a review for bike B if they've had a ride on B and likewise for user-to-user reviews. While it may seem simpler to only allow user X to review bike B a single time, we decided that every time a user rides a bike they will have the opportunity to write a new review. So, if user X rides bike B one time and then rides bike B again 2 weeks later, they can write a total of two reviews. We decided to allow multiple reviews to be written so that changes in the condition of the bike over time can be accounted for. Further, users can edit a review up to 2 weeks after their ride. However, after 2 weeks the review is locked to prevent users from giving inaccurate feedback. So, if user X rides bike B seven times over the course of a year and has a bad experience on the 7th ride, they will not be able to go back and change all their previous reviews in order to skew the ratings.

Should users be able to change their username and password?

While Fritter allowed usernames to be changed, since we are dealing with physical bikes worth a great deal of money, we decided that disallowing users from changing their usernames would help prevent fraud and make sure that users are more easily able to give accurate reviews. For example, if user A rents bike B to user Z, then user A changes their username, and later user Z goes to review A, Z might get confused if they initially rented the bike from a different username. However, since people frequently forget their passwords, and frequently changing

passwords is proven to make accounts more secure, we decided that changing an account's password will always be permitted.

How should the bike listings be displayed? With a list? With a map?

While we considered making listings list only, communicating locations is difficult. On the other hand, a map view is great for identifying spatial location, but can get cluttered if you try to show all the details on the map (especially if many bikes are close to each other). Thus, we decided to combine the two and show both a map view and list view side by side. This way, if a user clicks a pin on the map, the list will scroll to that listing so that the user can get more details. Likewise, if a user hovers over a specific listing, the corresponding pin on the map will be highlighted (with a different color) so that users can seamlessly transition between the two views and easily use the benefits of each view to improve efficiency and comprehension. That being said, because showing both views is not optimal small screens, we will incorporate a toggle feature that will allow you to hide the map view so that smaller screens don't get too cluttered.

Can a user delete their account?

While we considered enabling users to delete their own accounts, for the safety of all the users we have decided to disallow account deletion for the time being. First of all, we don't want reviews deleted along with accounts because these reviews themselves are public data and likely remain true after the original reviewer wants to delete their account. Second, if a user posts something inappropriate or fraudulent, we want to have as much data as possible to provide to law enforcement so that they can identify and find the perpetrator more easily. While storing account data on our platform does take up space on our servers, we value the safety of our users first and foremost.

How is the rating of a user and/or bike calculated?

While a typical way to do ratings is to just average all the ratings across all time, we want to encourage users to continue performing well in the future even if they already have a great rating. Further, bikes deteriorate with use and time, so it would be best to value more recent ratings higher than past ratings in order to account for the current state and condition of the bike. Thus, we will use a weighted moving average to determine ratings of both users and bikes.

Heuristic Evaluation (Jinny)

Visibility of system status

The “visibility of system status” heuristic rests on the idea of open and continuous communication. This heuristic can be fulfilled in two ways: persistent/always visible modules, and feedback that is only shown after the user has done something. The Wheel Deal fulfills this heuristic on both ends. Showing persistent state is exemplified by both the map toggle, whose appearance changes and persists based on the state of map visibility, and upon logging in, where the site switches the buttons “Sign Up” and “Log In” into “Account” and “Log Out” as indication that the user is logged in. We also use visual feedback to let people know their interactions with our system were acknowledged, as seen by the button color changes upon hover and press.

Match between system and the real world

The “match between system and the real world” heuristic says that the system should speak the user’s language and follow real-world conventions, making sure that the information appears in a logical order. Whereas we could use jargon to describe components in our system, we instead convey their usage using terms that are more direct, and that most people would recognize such as “Listing” in creating a new bike listing. Then, we ensure that our main action, reserving a bike presents all needed information in a logical order. On the home page, we show all potential bikes and the user can choose to view the listing for a single bike by clicking on the appropriate bike. This then takes them to the bike listing with a more comprehensive view of the bike details, which allows them to reserve after they input a reservation time period.

User control and freedom

The “user control and freedom” heuristic states that users often enter a state by mistake and need clearly marked “emergency exits” to leave their unwanted state (i.e. support undo and redo). The Wheel Deal addresses this heuristic in several ways. One way is in simply providing a “Home” button to users. The home page is the main endpoint by which key features and functions can be accessed, so people can always “return” to their main page of interest easily and quickly. We also provide “Cancel” options when users are filling out forms, or the option to “Delete” a bike as a sort of “undo” creation.

Consistency and standards

The “consistency and standards” heuristic states that systems should aim to maintain consistency and adhere to standards in existing frameworks. In this age, many people have used or at least seen marketplace-style websites. The Wheel Deal addresses this heuristic by adhering to such standards often seen in these kinds of websites, providing prices and locations for each of listings as well as filters to narrow down listings.

Error prevention

The “error prevention” heuristic aims to prevent problems from occurring on the side of the user by either having the system eliminate error-prone conditions altogether, or by having the system checking for these conditions and having users confirm the actions they’re about to take. The Wheel Deal demonstrates this principle most obviously in form submissions. Upon completion of a form, users are given the choice to either cancel or submit their form and the differences are highlighted by the buttons themselves (in terms of color). In this way, users can avoid the frustration and need to redo their previous work because the visuals helped them subtly get past these potential errors to begin with.

Recognition rather than recall

The “recognition rather than recall” heuristic emphasizes that recognition is superior to recall in user interface design because it provides more cues for the user to understand and efficiently use the system. The Wheel Deal’s user interface addresses this heuristic particularly well because all functions are contextualized with icons and/or self-explanatory words (e.g. in buttons), and items are grouped logically under their respective categories (e.g. all personal listings and their data are controllable by the user in their “Accounts” page).

Flexibility and efficiency of use

The “flexibility and efficiency of use” heuristic says that in good user interface design, flexible processes can be carried out in different ways, so that people can pick whichever method works well for them. The Wheel Deal offers a major accelerator in terms of functionality. For users who just want to rent a bike quickly and without having to think too much about details, the Wheel Deal offers users the choice to have us pick the bike for them. We also offer users the choice to choose location by picking a point on a map, rather than typing in the latitude and longitude of their bike’s location each time.

Aesthetic and minimalist design

The “aesthetic and minimalist design” heuristic ensures that users can focus on the key parts of the systems that they’re using (i.e. maximizing the signal-to-noise ratio). This also encompasses making sure that icons and images support the goals of the components they’re tied to. On the home page, the Wheel Deal avoids “adding noise” by only showing the most relevant information to all users. In other words, each listing emphasizes the bike’s rating and price, two of the most important and considered factors in choosing items off of a marketplace, with color and font styling. The user sees only the most relevant information to them on the home page (with respect to the bikes), and can find out more as necessary by clicking on the listing, which takes them to a different page that shows primarily the listing data.

Help users recognize, diagnose, and recover from errors

The error recognition and recovery heuristic consists of three steps: 1) inform the user when an error has occurred, 2) tell users what went wrong, and 3) offer users a solution. The Wheel Deal helps users address these steps by highlighting fields in red and giving them an

informative message about what went wrong under the field with the error. For example, if the user omits a username during sign up, the Wheel Deal outlines the username field in red and gives the user a concise and informative error message prompting them to input a username.

Help and documentation

The “help and documentation” heuristic emphasizes the importance of providing good documentation so users can help themselves carry out their task. At this point, the Wheel Deal is simple and straightforward enough that most users should be able to use the application without a problem. Therefore, we do not yet have a dedicated “Help Center.” However, we find that users can “help” themselves fairly easily because of the organization and direct fashion with which we present the features on the Wheel Deal.

Social/Ethical Design (Carlos)

Key Stakeholders:

- Bike riders
- Bike owners

As an application where users will interact with each other's real world property at specified locations, there are a variety of ethical concerns that arise that are common to ride-sharing apps and other dockless bike or scooter apps. These can be related to location data usage, fee structure, how to handle stealing or damages to the property in question, as well as more general concerns like what pictures can be uploaded for bikes or what content can go into reviews.

Location Data Usage:

A key concern arises around the use of location data related to the bikes. Making use of this data is necessary for preventing bikes from being stolen, coordinating and verifying rides, and improving rider experience. However, this data can easily be abused. The extent of the abuse could range from unwanted location aware ads to something very serious, such as kidnapping. For this reason, we do not collect real-time location data from the bikes. Location is only used when creating the bike and updated at points where it is essential, such as when a ride is starting or ending.

Stealing/Damaging Bikes:

The stealing or damaging of bikes belonging to users is another important ethical concern. Unlike many other bike-sharing apps where the bikes are owned by the company, in our case the bikes are owned by users, making it even more important to ensure they are protected. To address this, bike owners will submit the value of their bike and riders will agree to pay this amount if the bike not returned. On the side of damages, owners could report damages and submit repair costs, while riders would take pictures to prove they did not significantly damage the bike. In our case, pictures would be uploaded to an external photo sharing site and the link shared between parties through our platform. An electronic handshake would then take place, where both parties would agree that they are good with the state of the bike and agree to transfer responsibility over it.

Fee Structure:

A third important consideration would be the payment structure. In this case, we are going for a model where the full price of a ride is displayed upfront, similar to most ride-sharing apps. In this way, we will avoid users feeling that they were tricked or treated unethically when they discover that they paid more than they thought they would at the end of a ride. Obviously, certain fees, such as damage fees, will need to be added after the fact, but these will be the exception, not the rule.

Bike Pictures:

Yet another ethical consideration would be related to the bike pictures. Currently, users have the option to add bike pictures to their listings if they so desire via providing a link to an external photo. At the moment, however, there is no validation in place to ensure that the linked pictures are accurate, or even bikes at all. This creates a potential for abuse, since any image can currently be used as a bike picture. This means that not only can users post misleading or deceitful bike pictures, but they can also put other undesirable pictures on their bike listing for other users to see.

Additional Points

Availability Tracking:

In order to keep track of bike availability, we came up with the following schema. The smallest “slot” of time a user can rent a bike is 30 minutes. In our database, each bike would be represented by multiple rows in our database, each encoding the bike’s availability for each slot. This encoding can be done with a column in our database, with a bit set to 1 if the bike becomes taken for that slot, or 0 otherwise. When a bike owner creates a new listing for a bike, they may only make the bike available for a set number of slots. For example, if bike owner A were to make a listing for bike B, where B is available starting on January 1st, they can only make it available continuously for the next 2 months (ending on March 1st). Then, the database would have a row representing the first 30 minutes of January, for the second 30 minutes, etc. until March 1st. An alternative to this schema is to implement a linked-list-like data structure with NoSQL, but this may be nontrivially more difficult than the above method (when the above method would suit our purposes just fine).