}

Dream Team

## Team A

## 03/26/2021 - Milestone 3

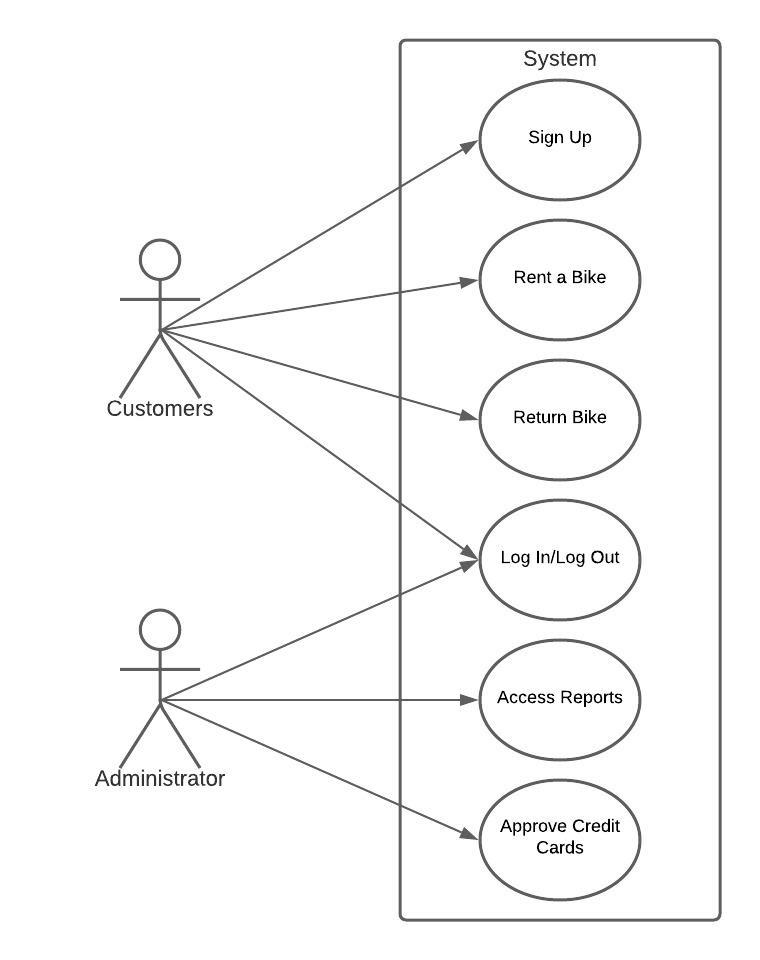
**Group Roles:**

1. **Janiece Campbell**: Creative Director, Flex Developer
2. **Ferris DeHart**: Frontend Developer, Interactive Testing Specialist
3. **Seth Richard**: Scrum Master, Backend Developer
4. **Ferol Schoonmaker**: Project Manager, Backend Developer
5. **Nick Settoon**: Frontend Developer, Database Specialist

**Contribution**

1. **Janiece Campbell**: 100%
2. **Ferris DeHart**: 100%
3. **Seth Richard**: 100%
4. **Ferol Schoonmaker**: 100%
5. **Nick Settoon**: 100%

**User Stories:**

1. The user will be prompted to sign into the website. The user will click on login if they have an account already. If the user does not have an account they will click sign up.
2. If the user selects login they will be given a username and password field to fill out. If the user fails to login after 3 tries they will be asked if they need to reset their password.
3. If the user selects to reset their password, an email will be sent to the email address associated with the user’s account and they will be given a temporary password. After using the temporary password the user will be prompted to set a new password.
4. If the user selects sign up they will be given a username, email, password, confirm password, and EULA field. The user will put the email they wish to associate the account with in the email field. If the email is already in use by another account on the website the user will be prompted to use a different email. The user will put their desired username into the username field. If the username is already being used by another account on the website the user will be prompted to enter in a new username. The user will put their desired password into the password field, they will then re-type the password in the confirm password field. If the two fields do not have identical contents the user will be prompted to retype the password into the confirm password field. The user will confirm that they agree to the EULA that confirms that they understand the fees associated with the service as outlined by the project customer.
5. When the user has created an account they will be given a name field, an address field, a birthdate field, a phone number field, and a credit card number field, cardholder field, c.c. expiration date field, a c.c. security number field and a billing address checkbox. The user will fill out the name field with their name, the address field with their address, the birthdate field with their birthday, the phone number field with their phone number, the user will then fill out the credit card number field with a credit card number, the cardholder field with the name associated with the credit card, and the expiration date field with the cards expiration date in mm/yyyy order. The user will put in the three numbers on the back of the credit card into the security number field. If the address the user listed in the address field is the same address associated with the credit card number the user will check the billing address checkbox. If the user does not check the billing address checkbox they will be prompted to fill out another address field with the address associated with the credit card.
6. When the user is signed in they will be able to select a button labeled ‘Rent Bike.’ If the user presses the ‘Rent Bike’ button they will be prompted to scan the barcode on the bike, this will then ask for confirmation and display the hourly cost associated with the rental and a statement that says, “If the bike is not returned, the customer’s credit card will be charged $500. If a bike is damaged, the customer will be charged $200 for the damage, depending on the damage.” The user can cancel the rental at any time, however if the user confirms the rental, the current timestamp will be tracked and sent to the database. Once the user confirms the bike will be released from the dock and they can begin biking. After confirming, there will be a prompt to rent another bike, which they can easily close.
7. Once the user replaces the bike in the rack, the rack locks the bike. Then the rack would tell the server to end the rental, which would notify the user that the rental period is over. The user would get an email receipt of the transaction.
8. Managers can log in from the same landing page as the client using a special, preset login. From the admin level site, they can choose to see/generate current and previous weekly reports. Current bike requests are also displayed, and the admin can choose to accept or deny based on card validity (or some other reason).

**Use cases:**

1. Administrator User:
   * Accesses weekly reports
   * Approves credit cards
2. Customer User:
   * Rent a bike
   * Log In
   * Log out
   * Sign up

**Project Backlog:**

1. Create Initial Web-App
2. Create a database to store user information, bike and dock information
   * Able to store private information (credit card information, passwords, and addresses)
   * Able to differentiate between administrators and customers.
   * Able to see if a bike is checked out, or damaged, and what dock it was last checked out and left at.
   * Able to see what bikes are available at a dock.
3. Add user logging in and registration
   * Existing users in the database can access their user profile
   * New users will be able to create a user profile on the website
4. Add Credit Card handling
   * New users will be prompted to add credit card data at sign up.
   * Credit card data will be verified after the user inputs it.
   * Existing users will be able to change their credit card data.
   * Credit card data will be verified after the user inputs it.
5. Add Bike Renting and Returning
   * Customers will be able to access the rent feature.
   * The user will select the dock they want to rent the bike from.
   * The user will be given a list of all available bikes at the dock.
   * The user will choose the bike they want to rent.
   * The user will be prompted with a reminder of the terms and conditions and fees associated with the renting process.
   * The bike will be set to a rented status and will not be available for rent by other users until it is returned, and is in ride-able condition.
   * The dock the bike was checked out from will be stored
   * When the user returns the bike they will input the dock they are returning too.
   * The dock’s bike inventory will be updated with the bike.
   * The user will be charged for the length of time the bike was out, and if the return dock is different from the checkout dock they will be charged the appropriate fee.
6. Handle damaged bikes
   * If the bike is found to be damaged the user will be charged the damage fee, and the bike will be unavailable for rent.
   * If the bike is not returned after a set period of time, the user will be flagged and charged for bike replacement.
7. Add a report system accessible by Mr. Dowling
   * The report system will be accessible by only administrator accounts
   * It will display the number of bikes rented in a specified frame of time.
   * It will display the number of bikes damaged in that frame of time.
   * It will display the number of users in the database.
   * It will display the revenue of bike rentals.
   * It will display the revenue of extra fees users have accrued.
   * It will display the combined total of those revenues.

**Backlog for Third Sprint:**

1. Create a handling for damaged bikes
2. Create an administrator account for Mr. Downling
3. Create a report feature for Mr. Downling
4. Create tests for the report feature
5. Run tests for the report feature to ensure it is working properly
6. Create tests for the damaged bikes
7. Run tests on the damaged bike feature to ensure it is working properly

**Test Cases**

Bike Renting and Returning

* 1. Test 1
     1. Input:
        1. bike number
        2. user account profile info
        3. current date and time
     2. Function under test: RentController.Submit()
     3. Expected Output:
        1. register a bike rental with the database
     4. Output: bike rental is logged in the database and bike is considered checked-out
  2. Test 2
     1. Input:
        1. bike number
        2. user account profile info
        3. current date and time
     2. Function under test: RentController.Return()
     3. Expected Output:
        1. inform database that bike has been returned
        2. give user receipt for rental
     4. Output: Return request is in the database, bike is available for future check-outs, price is given to the user.

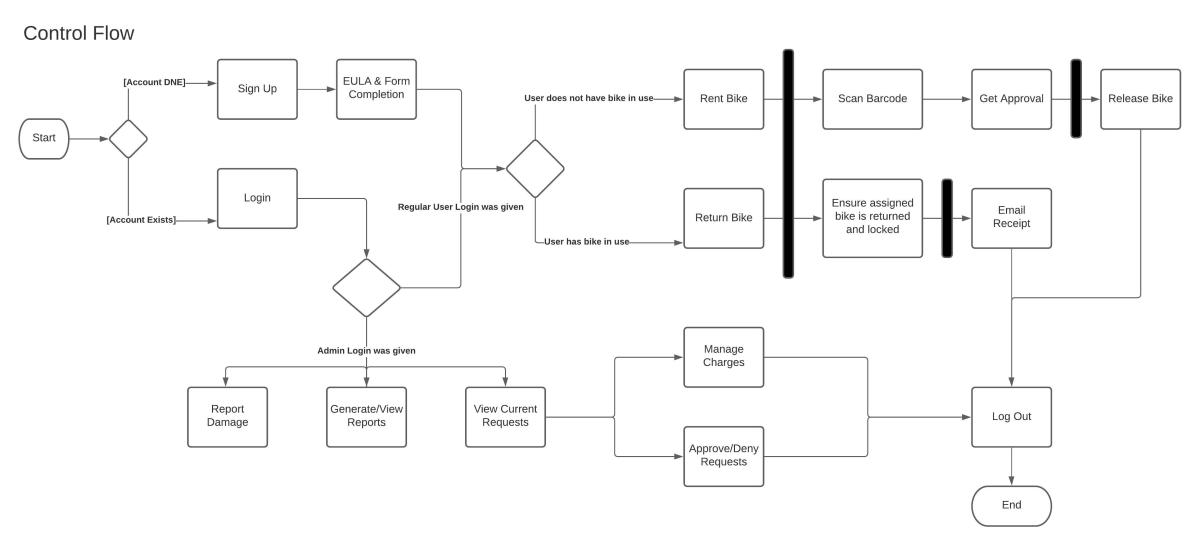
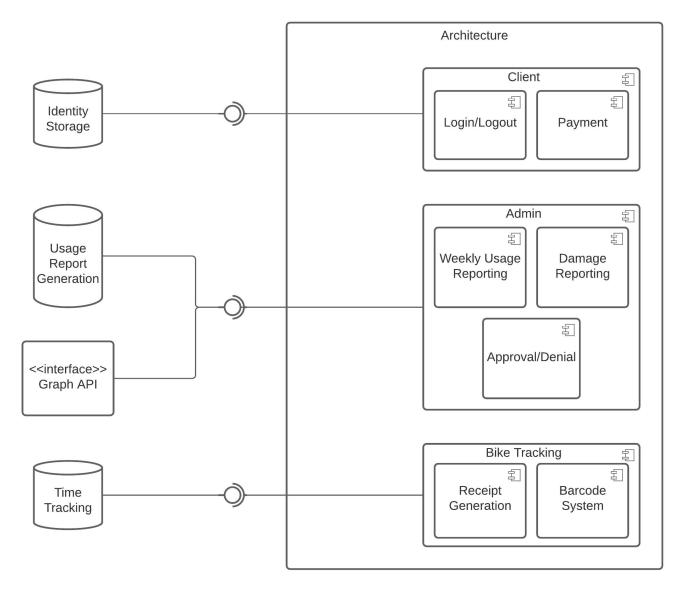
User login

1. Test 1
   1. Input:
      1. Valid Username
      2. And valid Password
   2. Function under test: ASP.NET Identity
   3. Expected Output:
      1. Successful login page
      2. User is considered signed into the account and able to access the rent and check-in features
   4. Output: User is signed in, and has access to all available features
2. Test 2
   1. Input:
      1. Invalid Username
      2. Or invalid Password
   2. Function under test: ASP.NET Identity
   3. Expected Output:
      1. Denied access/error notification
      2. Prompt to try logging in again with different credentials
   4. Output: User is denied access and is notified of the failed sign in attempt. User is prompted to repeat with proper credentials

User Sign-Up

1. Test 1
   1. Input:
      1. Email address
      2. Password of appropriate length, at least one uppercase letter, a symbol, and number
   2. Function under test: ASP.NET Identity
   3. Expected Output:
      1. Successful signup page
      2. User prompted to confirm email
   4. Output: User creates an account, is logged in, is prompted to confirm email, and account is added to the database
2. Test 2
   1. Input:
      1. Invalid email address
      2. Invalid password case
   2. Function Under Test: ASP.NET Identity
   3. Expected Output:
      1. Unsuccessful signup page
      2. User prompted to correct the exact invalid credential
   4. Output: User is prompted to enter in the credentials that match the system’s standards and is not allowed to create an account until proper credentials have been entered. The user is informed of what standards they failed to meet.

**Architecture Documents (UML)**



**Future Tests:**

Admin portal only accessible by admins

1. Test 1
   1. Input: Administrator credentials
   2. Function under test: ASP.NET Identity
   3. Expected Output:
      1. Access to admin overview page
   4. Output:
2. Test 2
   1. Input: non-administrator credentials
   2. Function under test: ASP.NET Identity
   3. Expected Output:
      1. Denied access/error page
   4. Output:

Damaged bikes

1. Input:
   1. bike number
2. Function under test:
3. Expected Output:
   1. Charge user damaged/missing bike fee
4. Output:

Credit Card handling

1. Input:
   1. user account
   2. user credit card information
2. Function under test:
3. Expected Output:
   1. payment receipt for user's rental
4. Output: