**CS673 Software Engineering** 

**Team 5 - GearOnTheGo**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Lazaro Perez | QA Leader | *Lazaro Perez* | 9/7/2023 |
| Ahnaf Tajwar | Security Leader | Ahnaf Tajwar | 9/7/2023 |
| Shajee Ur Rehman | Configuration Leader | Shajee ur Rehman | 9/7/2023 |
| Jian Song | Architecture and Design | *Jian Song* | 9/7/2023 |
| Samantha Mathis | Team Leader | *Samantha Mathis* | 9/7/2023 |
| Saahil Vashishta | Requirement Leader | Saahil Vashishta | 9/7/2023 |

**Revision history**

| Version | Author | Date | Change |
| --- | --- | --- | --- |
| 0.1 | Saahil Vashishta | 9/8/2023 | 1. [First Draft - Related Work](#_heading=h.30j0zll) 2. [First Draft - Functional Requirements](#_heading=h.1fob9te) (Essential, desirable and optional) |
| 0.2 | Ahnaf Tajwar | 9/9/2023 | 1. [First Draft - Non-functional Requirements (Security)](#_heading=h.1fob9te) |
| 0.3 | Lazaro Perez | 9/10/2023 | 1. [Update to Overview section.](#_heading=h.gjdgxs) |
| 0.4 | Samantha Mathis | 9/10/2023 | [Management Plan: Objectives and Priorities](#_heading=h.d860n9fpce6c) |
| 0.5 | Jian Song | 9/10/2023 | 1. [Added more on Related Work](#_heading=h.30j0zll) 2. [Update on Configuration Management Plan](#_heading=h.1t3h5sf) |
| 0.6 | Lazaro Perez | 9/10/2023 | 1. [Updated Quality Assurance Section](#_heading=h.17dp8vu) |
| 0.7 | Samantha Mathis | 9/10/2023 | 1. [Management Plan: Risk Management](#_heading=h.3znysh7) 2. [Management Plan: Timeline](#_heading=h.3dy6vkm) |
| 0.8 | Saahil Vashishta | 9/11/2023 | Incorporated feedback comments posted by Shajee, added team logo and replaced instances of terms with Abbreviations as defined in [Glossary.](#_heading=h.44sinio) |
| 1.0 | Samantha Mathis | 9/12/2023 | Final Version for Iteration 0 |
| 1.1 | Saahil Vashishta | 9/24/2023 | 1. [Updating Deployment Plan](#_heading=h.2s8eyo1) 2. [Updating References](#_heading=h.1ksv4uv) 3. [Updating Glossary](#_heading=h.44sinio) |
| 1.2 | Shajee Ur Rehman | 9/25/2023 | 1. Slight changes to formatting of document 2. [Added 0008 ‘Checkout and Payment’ to essential features.](#_heading=h.1fob9te) 3. [Added 0017 ‘Stripe Secure Payment’ to optional features.](#_heading=h.1fob9te) |
| 1.3 | Saahil Vashishta | 10/09/2023 | 1. [Deprecating requirements that are out of scope](#_heading=h.1fob9te) 2. [Finalizing scope of requirements and improvement opportunities](#_heading=h.1fob9te) |
| 1.4 | Shajee Ur Rehman | 10/10/2023 | 1. Deprecating requirements that are no longer desired. |
| 1.5 | Lazaro Perezz | 10/10/2023 | 1. Updated the Quality Assurance Testing and Defect Management sections |
| 2.0 | Saahil Vashishta | 10/10/2023 | Final Version for Iteration 2 |
| 2.1 | Saahil Vashishta | 10/14/2023 | 1. [Addition of Use Case Diagram](#_heading=h.1fob9te) 2. [Addition of Cumulative Flow Diagram](#_heading=h.1fob9te) 3. [Requirement Analysis](#_heading=h.1fob9te) 4. [Updated Deployment Plan](#_heading=h.2s8eyo1) |
| 3.0 | Saahil Vashishta | 10/15/2023 | Final Version for Iteration 3 |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# Overview

Our product is a recreational equipment rental service web application. Think of a craigslist but for equipment rentals. The motivation to create a website that allows people to rent items for lease, stems from the need for some people to only use these items for short time periods, meanwhile someone in the city or state where said people are visiting for the purposes of doing their intended recreational activity may have the equipment ready to use and is simply sitting in their houses not being utilized. Our vision is to align people who are in need of these recreational equipment, examples include(canoes, kayaks, mountain bikes, boats, jet skis etc), with people who have the equipment readily available for others to use for a specified time. The further need also comes from Individuals that are in need of renting items that do not wish to own or take full liability of damage. The functionality of the site is to allow users to choose items that are posted by individuals in their vicinity who need these items. The possible technology stack we will need for this project will be as follows - Back End - Python/Flask with a Postgresql Database, followed by a front end website built in React/JS.

# Related Work

There are a few web applications and mobile applications that are somewhat similar to the web application we are planning to develop but there are critical differences that make our idea unique. The following are some of the current ideas out there:

1. [**KitLender:**](https://www.kitlender.com/?season=camp&gad=1&gclid=CjwKCAjwjOunBhB4EiwA94JWsLN9FozSJBALmummeIuJDBriTEqoeciBcfz0DNzja0LKMoiFRgVnJBoC-AAQAvD_BwE) Kit Lender is a web based application that allows users to rent camping gear, hiking gear and other recreational gear and equipment directly from the website no matter where the user is. Products can be searched from a comprehensive catalog and rented out for a predetermined amount of time. The gear is charged at a daily rate.

Similarities:

1. The web application allows the user to source recreational equipment to rent, which is what our application plans to provide as well.
2. It provides means to filter and search through the inventory, which is a feature we would like to implement in our application as well.

Differences:

1. The inventory for Kit Lender is maintained by Kitlender and branded by them as well. They stockpile this equipment. Our application allows the user to rent equipment that belongs to other equipment users in the area, much like air bnb allows users to rent pre-owned homes. We do not plan to maintain an inventory.
2. Since the equipment is not maintained by us and owned by someone else, our application has the capability to provide competitive pricing (no overheads for logistics, storage, purchasing equipment)
3. Kit Lender delivers equipment to the doorstep, thus adding logistics to their business whereas our application allows the user to directly collect the equipment from the owners, thus providing instant access rather than waiting for deliveries. This way, the user can rent on the same day they decide to source rental equipment.
4. [**Turo:**](https://turo.com/us/en/search?country=US&defaultZoomLevel=11&endDate=09%2F13%2F2023&endTime=10%3A00&isMapSearch=false&itemsPerPage=200&latitude=47.6061389&location=Seattle%2C%20WA&locationType=CITY&longitude=-122.33284809999998&pickupType=ALL&region=WA&sortType=RELEVANCE&startDate=09%2F10%2F2023&startTime=10%3A00&useDefaultMaximumDistance=true&utm_campaign=gs%3Abr%3Aexact%3Awd%3Abrand%2014497529174&utm_medium=cpc&utm_source=google&utm_term=gs%3Abr%3Aexact%3Awd%3Abrand-gen%20kwd-479795307%20667602463099%20126742830197) Turo is a web based application as well as a phone application that allows the user to rent vehicles that are owned by people. Vehicles can be searched based on availability in a particular area. The vehicles are reserved and then collected by the user from specific addresses for a daily/weekly/monthly rate. After use, the user needs to drop the vehicle back to where they collected it from or to another designated spot as provided by the vehicle owner.

Similarities:

1. The web application allows the user to source recreational equipment to rent, which is what our application plans to provide as well.
2. The user can search for available equipment in a given area

Differences:

1. Turo is a similar concept but it is exclusively for renting vehicles, it does not allow the user to rent recreational equipment.
2. Turo is both a web application and a mobile application, at this point to avoid complexity, we will be sticking to developing a web based application.
3. [**REI:**](https://www.rei.com/stores/rentals)REI is a brick and mortar store in addition to being a web based application, that allows the user to locate stores based on their location, choose recreational equipment and rent it for a predetermined duration. REI maintains its own products as a manufacturer and rents out REI branded equipment and in addition to that, it also rents equipment from other major brands that it directly purchases as a distributor.

Similarities:

1. The web application allows the user to view recreational equipment to rent, which is a feature our application plans to provide as well.
2. It provides means to filter and search through the inventory, which is a feature we would like to implement in our application as well.

Differences:

1. The inventory is maintained and to some extent manufactured by REI. They brand and distribute from a brick and mortar store whereas our application just connects passionate renters to owners who can profit from renting their equipment.
2. Users can only rent from pre-existing REI stores whereas our application allows the flexibility to rent from anyone hosting recreational equipment for rental.
3. Since the equipment is not maintained by us and owned by someone else, our application has the capability to provide competitive pricing (no overheads for logistics, storage, purchasing equipment)
4. [**Zipcar**](https://www.zipcar.com/)**:** Zipcar is a web/mobile based app that allows the user to rent vehicles short term - a key difference from Turo is that Zipcar is a running platform and all vehicles are held and owned by it instead of individuals. Vehicles can be searched based on availability in a particular area. The vehicles are reserved and then collected by the user from specific addresses for an hourly/daily rate After use, the user needs to drop the vehicle back to where they collected it from or to another designated spot as provided by the vehicle owner.

Similarities:

1. The web application allows the user to look for recreational equipment at desired locations,
2. The rent can be flexible for short trips/hikes.

Differences:

1. Zipcar is not running in a light asset mode, which means that if we want to create an Airbnb/Uber like app that just connects the dots among people instead of distributing equipment, this might not fit.
2. Similar to the point mentioned above, maintenance and ownership is also aligned with the app, not people actually owning the equipment.
3. [**Uber**](https://www.uber.com/us/en/s/d/kochab/?ad_id=609972022297&adg_id=138556263253&campaign_id=17728442566&cre=609972022297&dev=c&dev_m=&fi_id=&gclid=Cj0KCQjw0vWnBhC6ARIsAJpJM6eksNEx_Lw4VCKgSjyRHj6AxhHJmTlM2OrSB3kHRNeCT75RvNh2Oi4aAm-jEALw_wcB&gclsrc=aw.ds&kw=uber&kwid=kwd-12633382&match=b&net=g&placement=&tar=&utm_campaign=CM2179741-search-google-brand_1_6_US-Boston_o-d_web_acq_cpc_en_T1_Generic_BM_uber_kwd-12633382_609972022297_138556263253_b_c&utm_source=AdWords_Brand)**:** Uber is a web/mobile based app that mainly provides ride-hailing services. In the app the user does not care about locations because the location services are automatically enabled. What users need to do is just call a ride for a fee determined based on riding time and distance.

Similarities:

1. The only action required from the user side is to look for needed recreational equipment and the app will take care of the rest by routing to the nearest people that have those on shelf to lease. This is ideal for people visiting another region.
2. The fee structure makes more sense in a more complex setup - both renting time and distance counts in this case and the equipment should be delivered by the owner.

Differences:

1. Of course Uber focuses on ride hailing, which is an essentially different business model. Only the algorithm side is similar to our app.
2. We need to figure out whether to include distance as a parameter as well as determine whether owners should deliver the equipment - if not, then the Uber model might not fit in this case.

# Proposed High level Requirements

* 1. Functional Requirements
     1. **Essential Features**

| **Requirement ID** | **Requirement Description** | **Estimated Time (hours)** | **Status** |
| --- | --- | --- | --- |
| 0001 | User Registration: A user shall be able to register with the web based application with the following information:   1. User Name AND 2. User Date of Birth AND 3. ~~User Phone number AND~~ 4. ~~Local Address AND~~ 5. User email ID OR User social media profile | 8-10 | Implemented |
| 0002 | User Login: A user shall be able to login with the credentials provided at user registration with password authentication. | 3-4 | Implemented |
| 0003 | Search Reservation Availability: A user shall be able to search for appropriate reservations using the following parameters:   1. Area of reservation OR Zip Code AND 2. Start Date of Reservation AND 3. End Date of Reservation AND 4. Start Time of Reservation AND 5. End Time of Reservation AND 6. Equipment required | 8-10 | Implemented |
| 0004 | Make reservation: A user shall be able to choose, reserve and pay for the equipment desired from the search results.  ~~Note:~~  *~~Acceptance and Delivery Timeframe: A time frame of xyz days needs to be adhered to for the HU to a) accept a reservation b) deliver the goods or face a penalty on their account because these rentals will be on the fly and if someone is traveling or found a more~~ ~~proactive host, they would prefer that their equipment reaches them in their desired dates.~~* | 6-8 | Implemented |
| 0005 | Host Equipment: A User shall be able to host the equipment using the following parameters:   1. Area of Equipment OR Zip Code AND 2. Start Date of Equipment Availability AND 3. End Date of Equipment Availability AND 4. Daily Price of Equipment AND 5. Description of Equipment | 8-10 | Implemented |
| ~~0006~~ | ~~Email Notification: The following shall get an email notification after a reservation has been made:~~   1. ~~RU AND~~ 2. ~~HU AND~~ 3. ~~RU shall receive another email once the reservation has been accepted.~~   ~~Note:~~  *~~Reminder Email Notification: HU receives an email reminder 48 h before the reservation Start Time to dispatch the equipment if they have not already. This can be determined by the Host clicking on an icon at their end which indicates that the equipment is out for delivery.~~*  *~~Failure To Comply Email Notification: Host receives an email if their equipment has not been delivered before the reservation Start Time and that their rental fee has been reduced by xyz %.~~* | ~~2-3~~ | Deprecated |
| 0007 | Comprehensible and easy to navigate UI: The web based application shall have a non-complex and easy to navigate user interface for all users | 6-8 | Implemented |
| 0008 | Checkout and Payment: Upon making a reservation request; A Renting User shall proceed to a Checkout Page where they can make a payment to the Hosting User. The Checkout Page shall require:   1. Name on Card AND 2. Billing Address AND 3. Card Number, valid through, CVC AND 4. Short message detailing reservation experience   The Checkout Page shall access the price from the reservation database and submit the payment to a payment info database, but not before validating the Card Number.  It will then route the user to a successful payment accordingly. | 12-14 | Implemented |

**Desirable Features (the nice features that you really want to have too):**

| **Requirement ID** | **Requirement Description** | **Estimated Time (hours)** | **Status** |
| --- | --- | --- | --- |
| ~~0008~~ | ~~Use Current Location: The application shall have the capability to pull user’s current location to auto fill for location parameter on user selection of an option named “Current Location”.~~ | ~~2-3~~ | Deprecated |
| 0009 | Search Reservation Availability with Price filter: A user shall be able to search for appropriate reservations, filtered out based on a user defined minimum daily price AND a user defined maximum daily price. | 2-3 | Implemented |
| ~~0010~~ | ~~Host Equipment with images: A User shall be able to host the equipment and be able to provide 1 or multiple images of the equipment being rented.~~ | ~~8-10~~ | Deprecated |
| 0011 | Reservation Rating: The HU shall be able to rate the RU after each reservation AND the RU shall be able to rate the HU after each reservation.  ~~Note:~~  *~~These will become available 48h after the reservation End Time has completed and the User must return the equipment within that time frame.~~* | 2-3 | Implemented |

* + 1. **Optional Features (additional cool features that you want to have if there is time):**

| **Requirement ID** | **Requirement Description** | **Estimated Time (hours)** | **Status** |
| --- | --- | --- | --- |
| ~~0012~~ | ~~Chatbot: The Web application shall have an automated AI chat bot to solve any questions the user has.~~ | ~~5-7~~ | Deprecated |
| ~~0013~~ | ~~FAQ: The web application shall have a section for Frequently asked questions with answers.~~ | ~~1-2~~ | Deprecated |
| 0014 | Host and Renter profile page: The RU and HU shall have public OR private profiles with the following information:   1. Photo of the user AND 2. Review comments from previous reservations AND 3. Optional Text as Bio | 8-10 | Implemented |
| ~~0015~~ | ~~Insurance purchase option: The HU shall have the ability to purchase equipment insurance through the web application portal.~~ | ~~2-3~~ | Deprecated |
| 0016 | About Us: The Web Application shall have an About Us page with roles and 1 fun fact for each of the team members involved in the project. | 1 | Implemented |
| ~~0017~~ | ~~Stripe Secure Payment: The Checkout and Payment process may include the Stripe Secure Payment processing feature.~~ | ~~6-8~~ | Deprecated |

* 1. Nonfunctional Requirements
     1. Security requirements

| **Requirement ID** | **Requirement Description** | **Estimated Time (hours)** | **Status** |
| --- | --- | --- | --- |
| ~~0017~~ | ~~Forgot Password?: If the user forgets their password, an email verification and password reset process can be added~~ | ~~5-7~~ | Deprecated |
| ~~0018~~ | ~~ID Verification: User provides government issued ID when signing up for an account to verify they are a real person~~ | ~~4-5~~ | Deprecated |
| 0019 | Hash password: User created password is stored as a hash value | 2-4 | Implemented |
| ~~0020~~ | ~~Secure Payment: Payment transfer is secure and verified. Payment gateway needs to be PCI-DSS compliant~~ | ~~6-7~~ | Deprecated |
| ~~0021~~ | ~~SQL Injection Prevention: Attackers may inject malicious code into an application's database through user input fields, gaining unauthorized access to sensitive data stored in the database. To prevent, implement these below.~~  ~~Validate user input~~  ~~Use output encoding, which involves converting special characters such as < and > into their HTML entity equivalents, to prevent them from being interpreted as HTML code.~~  ~~Use prepared statements, parameterized queries, or stored procedures instead of dynamic SQL whenever possible.~~ | ~~6-7~~ | Deprecated |
| ~~0022~~ | ~~Scam Prevention:~~  ~~Possibly have the money being paid securely to the website (a bank account of the company) first and is only further routed to the HU’s bank account after the RU clicks to confirm that they have received the equipment in order to prevent scammers from creating profiles, uploading pictures of equipment and receiving payments which are then transferred elsewhere and difficult to retrieve.~~  ~~This would include a system for the money to be refunded after a simple investigation process.~~ | ~~10-30~~ | Deprecated |

Requirements that may not be fulfilled due to time constraints but may be considered for future development can be seen below:

| **Requirement ID** | **Requirement Description** | **Estimated Time (hours)** |
| --- | --- | --- |
| 0006 | Email Notification: The following shall get an email notification after a reservation has been made:   1. RU AND 2. HU AND 3. RU shall receive another email once the reservation has been accepted.   Note:  *Reminder Email Notification: HU receives an email reminder 48 h before the reservation Start Time to dispatch the equipment if they have not already. This can be determined by the Host clicking on an icon at their end which indicates that the equipment is out for delivery.*  *Failure To Comply Email Notification: Host receives an email if their equipment has not been delivered before the reservation Start Time and that their rental fee has been reduced by xyz %.* | 2-3 |
| 0008 | Use Current Location: The application shall have the capability to pull user’s current location to auto fill for location parameter on user selection of an option named “Current Location”. | 2-3 |
| 0010 | Host Equipment with images: A User shall be able to host the equipment and be able to provide 1 or multiple images of the equipment being rented. | 8-10 |
| 0012 | Chatbot: The Web application shall have an automated AI chat bot to solve any questions the user has. | 5-7 |
| 0013 | FAQ: The web application shall have a section for Frequently asked questions with answers. | 1-2 |
| 0015 | Insurance purchase option: The HU shall have the ability to purchase equipment insurance through the web application portal. | 2-3 |
| 0017.1 | Stripe Secure Payment: The Checkout and Payment process may include the Stripe Secure Payment processing feature. | 6-8 |
| 0017.2 | Forgot Password?: If the user forgets their password, an email verification and password reset process can be added | 5-7 |
| 0018 | ID Verification: User provides government issued ID when signing up for an account to verify they are a real person | 4-5 |
| 0020 | Secure Payment: Payment transfer is secure and verified. Payment gateway needs to be PCI-DSS compliant | 6-7 |
| 0021 | SQL Injection Prevention: Attackers may inject malicious code into an application's database through user input fields, gaining unauthorized access to sensitive data stored in the database. To prevent, implement these below.  Validate user input  Use output encoding, which involves converting special characters such as < and > into their HTML entity equivalents, to prevent them from being interpreted as HTML code.  Use prepared statements, parameterized queries, or stored procedures instead of dynamic SQL whenever possible. | 6-7 |
| 0022 | Scam Prevention:  Possibly have the money being paid securely to the website (a bank account of the company) first and is only further routed to the HU’s bank account after the RU clicks to confirm that they have received the equipment in order to prevent scammers from creating profiles, uploading pictures of equipment and receiving payments which are then transferred elsewhere and difficult to retrieve.  This would include a system for the money to be refunded after a simple investigation process. | 10-30 |

**3.1 Use Case Diagram:**

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

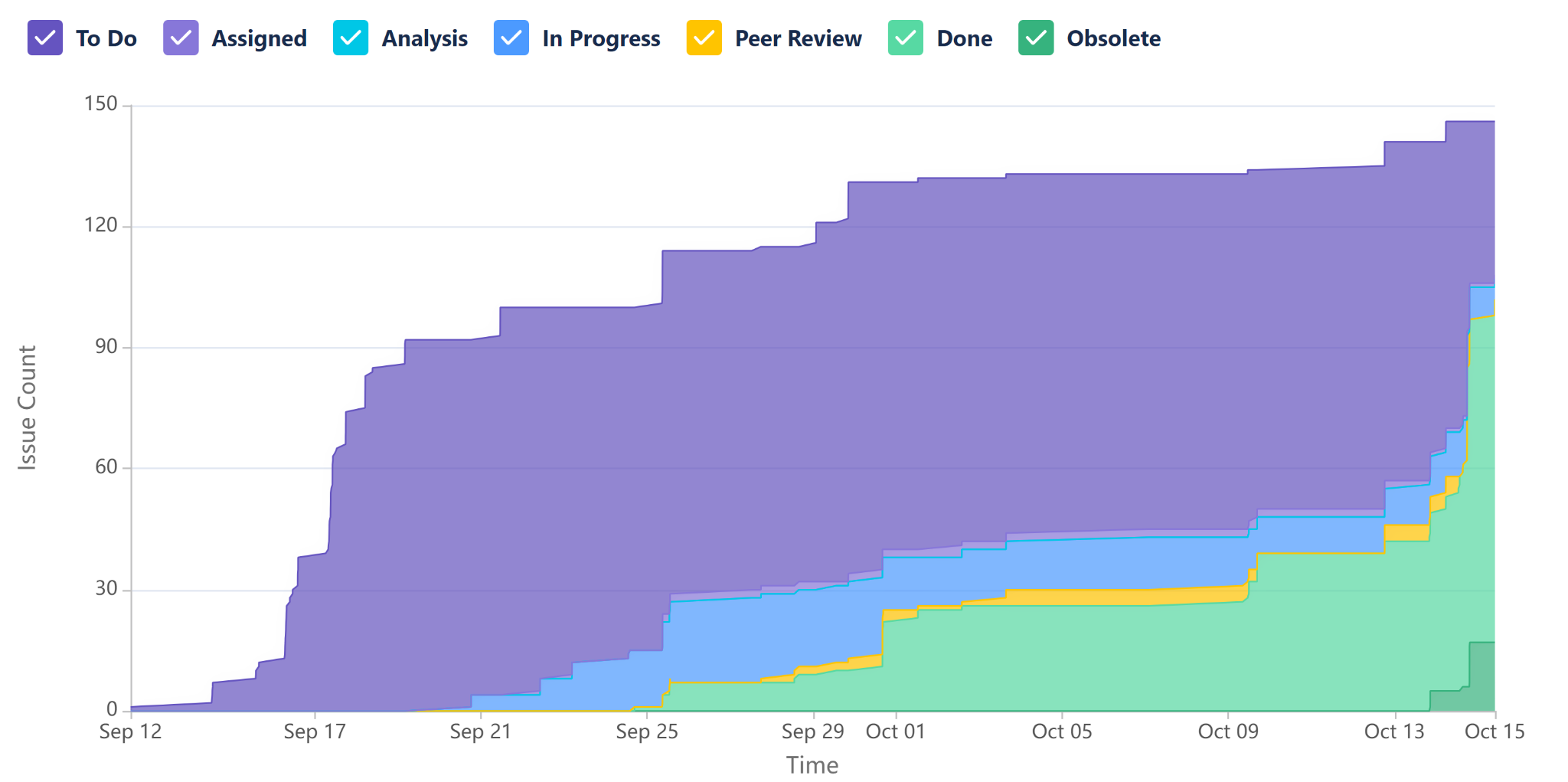
# 

# 

# 

# 

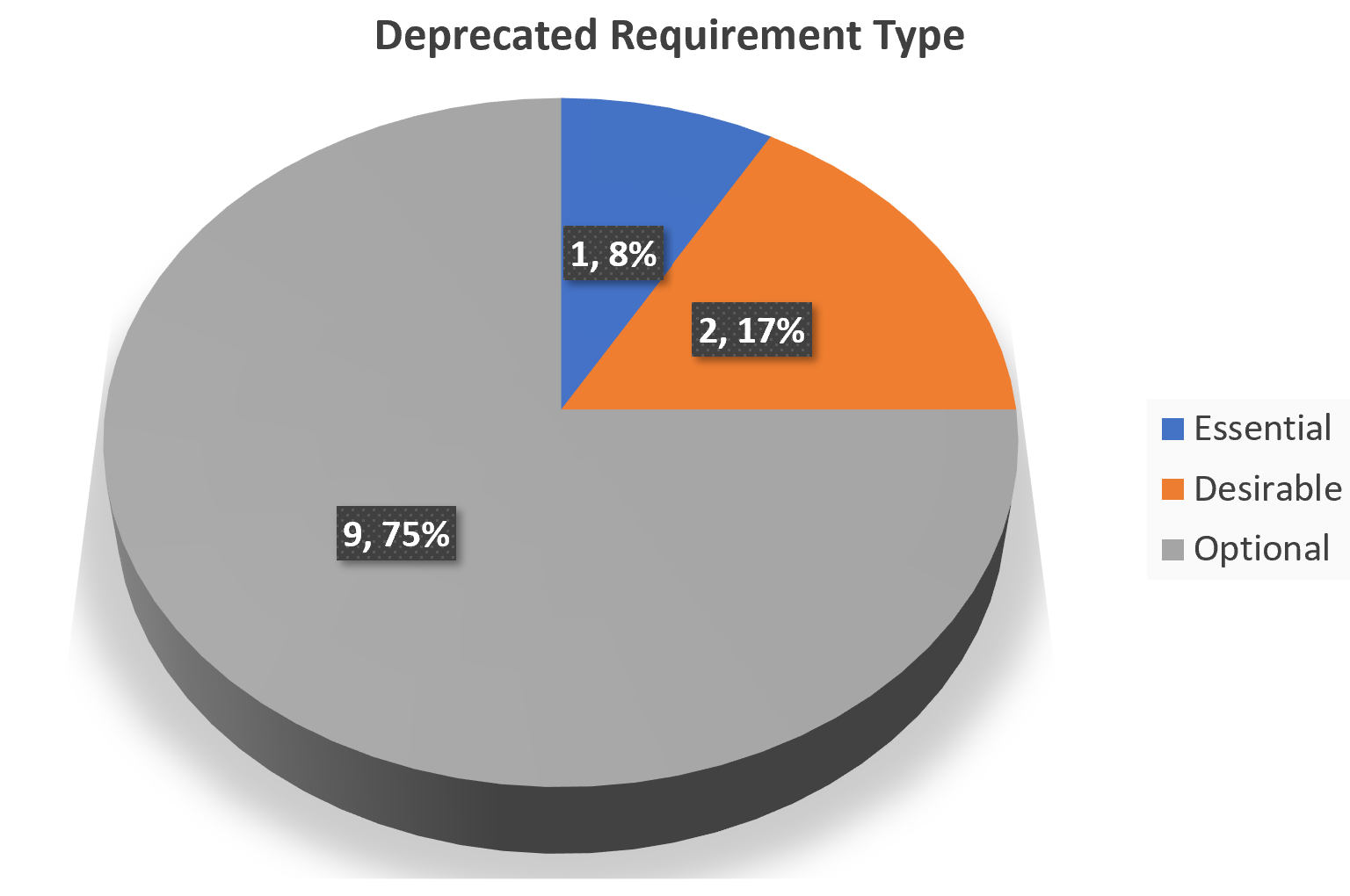
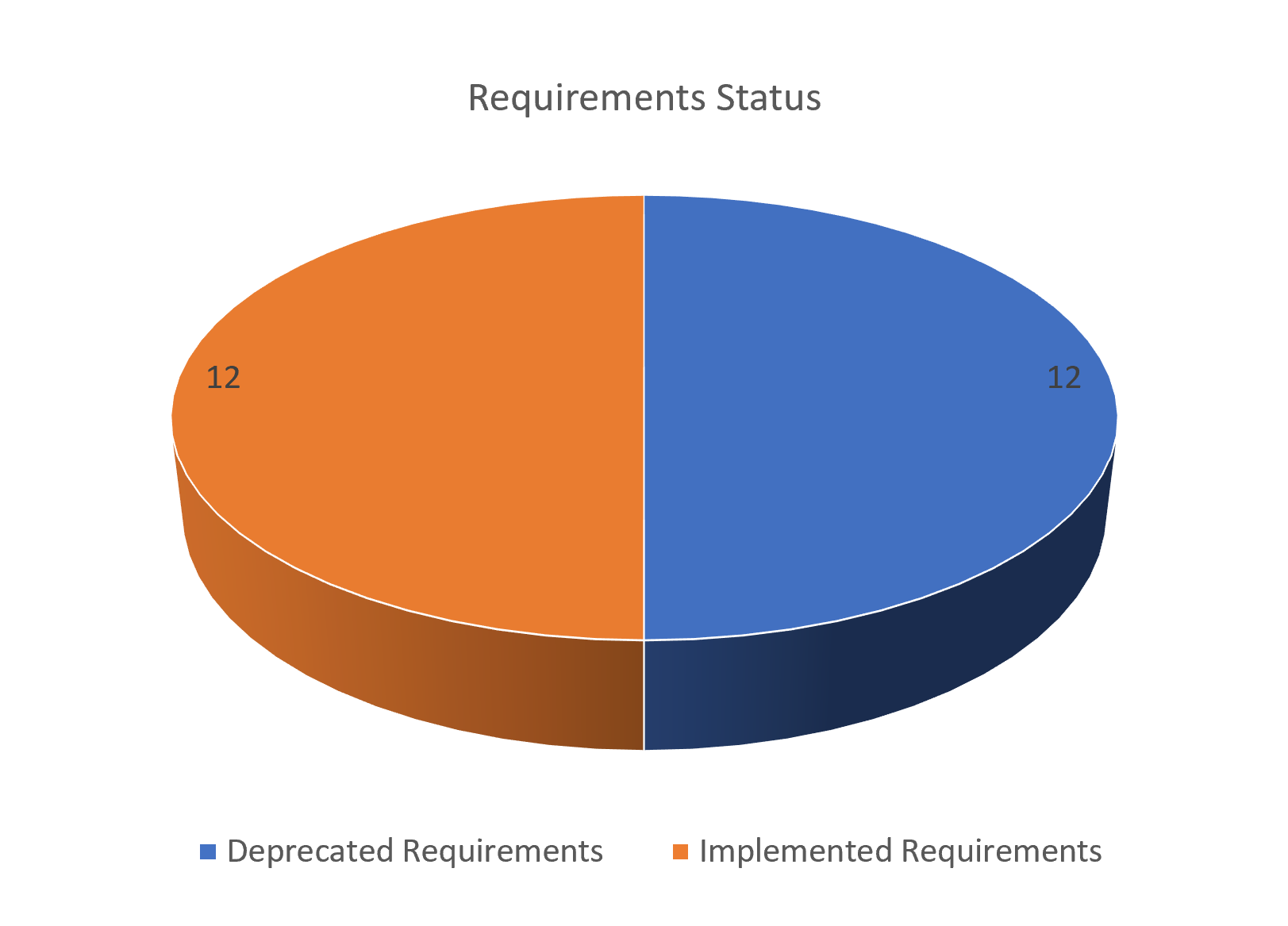
**3.2 Cumulative flow diagram**

****

* The cumulative flow diagram on the right shows the progression of tasks over time
* The trend we can see through the sprints is that the team was consistent about keeping the stories updated.
* A lot of the work was finished in the last sprint so from the pattern it can be seen that the last 2 sprints were extremely work intensive
* The gap between To do tasks and done tasks shows epics that the team took over but couldn’t finish due to the scope of this project and time constraints

**3.3 Requirement Analysis**

* There was a 50-50 split of requirements that were successfully implemented and the ones that were deprecated due to scope and time constraints.
* Based on the original scope, of the deprecated requirements:
  + Only 8% of the requirements were essential (1 requirement)
  + Only 17% of the requirements were desirable
  + A whopping 75% of the requirements were optional (9 requirements)

****

# Management Plan

## Objectives and Priorities

* + 1. Login Feature
    2. Database
       1. Users
       2. Items for rent
    3. Search for Items
    4. View your rentals
    5. View your items
    6. Checkout
       1. Timeframe
       2. Payment
       3. Confirmation
    7. Automatic returns after the end return date

## Risk Management (need to be updated constantly)

The highest priority risks that we decided were Not enough time for testing and not familiar with the programming language. We decided these were high priority because we think they are more likely to happen then some of the other risks. A few of us are still new to flask so it'll take some time to learn and get the hang of the framework. Testing is a key component of everything, without adequate testing we won’t know if later on we will encounter a bug or if all of our features work the same when they are put together. Some of the lowest priority risks were Improper planning, Duplicate work, Loss of a member, and improper management skills. This was decided because even though these are big risks that could potentially put us behind schedule we all agree it will be easy to avoid these risks. To avoid these risks we plan on open communication, weekly meetings, Tracking and assigning work.. All of the risks are detailed out in the risk management spreadsheet, listing the priority, the impact, detailed plan and execution summary.

**Risk Management Sheet Link:** <https://docs.google.com/spreadsheets/d/1xgI42GHNuiv2reJzhAqoGklrghc5D8CY/edit?usp=sharing&ouid=115670479894087574008&rtpof=true&sd=true>

## Timeline

| Iteration | Functional Requirements(Essential/Disable/Option) | Requirement Number | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- | --- |
| 1 (9/12 - 9/26) |  |  |  |  |
|  | User registration | 001 | A user shall be able to register with the web based application with information, being stored in a database | 8-10 |
|  | Login | 002 | A user shall be able to login with the credentials provided at user registration with password authentication. | 3-4 |
|  | Hosting Equipment | 005 | A User shall be able to host the equipment using parameters | 8-10 |
|  | Comprehensible and easy to navigate UI | 007 | The web based application shall have a non-complex and easy to navigate user interface for all users | 6-8 |
| 2 (9/26 - 10/10) |  |  |  |  |
|  | Make Reservation | 004 | A user shall be able to choose, reserve and pay for the equipment desired from the search results. | 4-6 |
|  | Search Features | 009 | A user shall be able to search for appropriate reservations, filtered out based on a user defined minimum daily price AND a user defined maximum daily price. | 8-10 |
|  | Profiles | 0014 | The RU and the HU shall have public OR private profiles with information about them | 8-10 |
| 3 (10/10 - 10/17) |  |  |  |  |
|  | ChatGPT | 0012 | The Web application shall have an automated AI chat bot to solve any questions the user has. | 5-7 |
|  | About us | 0016 | The Web Application shall have an About Us page with roles and 1 fun fact for each of the team members involved in the project. | 1 |
|  | FAQ | 0013 | The web application shall have a section for Frequently asked questions with answers. | 1-2 |

# Configuration Management Plan

## Tools

(In this project, we will use Git and Github as the version control tools. Please also specify any other tools to be used, e.g. IDE tools, CI/CD tools, container tools, SAST or DAST tools, and any other DevOps tools)

* All team members agreed to use Git and Github desktop as the version control tools.
* In terms of IDE, VS Code is chosen across the team.
* The team will use Jira to track progress.
* The team will use Lucid Chart for UML diagrams.
* The team will use Postman for API testing.
* Determination on use of other tools will come as the team progresses the project.
  1. Code Commit Guideline and Git Branching Strategy  
     (Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies: <https://www.flagship.io/git-branching-strategies/>

### The team will be using GitHub Flow branching strategy - starting off with the main branch then everyone will create a feature branch that stems directly from the main branch and work on their own. In the end the codes will be merged by pull requests from feature branches into the main branch. The feature branch is then deleted.

Pull request will be used whenever there is a merge from feature branches - all team members will be reviewers and it would require 2 approvals out of 5 reviewers to complete the merge.

## Deployment Plan

(If you plan to deploy your application (e.g. your web application), briefly describe how you plan to deploy your application).

The planned deployment for Gear To Go involves hosting the database on [Elephant SQL](https://www.elephantsql.com/) and the front end for the web application to be hosted on [Netlify](https://www.netlify.com/). As of Iteration 1, the code being hosted on Github, can be tested locally while updating the database on Elephant SQL in real time. We plan to provide Netlify access to our github repository for the front end.

**Database Deployment Plan:**

1. We created a single account on Elephant SQL and opted for the free plan which allows hosting 20 MB of data in tables and allows 5 concurrent users at a time.
2. An instance named Equipment was created on Elephant SQL.
3. All team members were invited as contributors and given access to the instance.
4. Tables were added to the database.
5. Connections to the database were made in different app.py branches.
6. Data persistence was checked.

**Front End plus Backend Deployment Plan:**

1. Create an account on Heroku.
2. Use the Sites option in the dashboard and opt for ‘new site’ from Git.
3. Select the Github repository.
4. Configure build settings (build command and output directory)
5. Review front end code and check environment variables.
6. Review Backend code
7. Trigger deployment.

**Post Deployment:**

1. Scaling of the database would need to be done based on required traffic.
2. Documentation would need to be maintained.
3. Frequent testing will need to be done to maintain the database and web application front end.

Link for the website can be found here: <https://gearonthego-52bc9f57a8cd.herokuapp.com/#/>

# Quality Assurance Plan

## Metrics

| Metric Name | Description |
| --- | --- |
| DB Tables | Database table structures that include the necessary attributes for each intended table. |
| API Connectivity | This metric is used to ensure that the API’s created in the backend are functioning and thoroughly tested for each use case and user story made |
| Code Coverage | How much of the code can be tested to ensure that bugs do not persist within the production of the software |
| Code Duplication | Metric to measure the use of redundant code in the project. |
| Test Pass Rate | This metric would be used to measure the amount of successful passing test cases while testing the Database as well as the functionality of the front end UI |

* 1. Coding Standard

**Code Formatting -** ensuring the code base is using the standards established by the team. This will help with **enhanced code maintainability**, code consistency, as well as a reduction in the likelihood of bugs during any coding updates.

**Variables and Functions -** Getting proper variable names and function names based on the item and functionality of the variable and functions. **Using concise and expressive variable names make the code more readable and understandable**.

**Design Patterns -** Using proper design patterns to implement for the **website functionality**.

**Documentation and Comments -** Coding Comments and Documentation used in the code base.

## Code Review Process

Each team **member will contribute to the code base**, and a formal review conducted by the Architecture and Design, Quality Assurance, and Configuration Lead will oversee that best practices are being followed in the code base.

## Testing

**Postman -**  will be used for API testing

**Unit Testing** - Done by each developer/team member

Manual Testing - Done by each developer/team member

PyTest- Python testing

Selenium - Web Application testing which is also being done by each developer/team member.

## Defect Management

**Github Issue Tracker will be used to track any defects** that are seen in the code base. As well as tracking these in **Jira and the Progress Tracker** based on the team member who is working and fixing the defects. We are also doing this in our coding reviews when there are any merge conflicts using the Github UI.

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

1. Related work:
   1. KitLender: <https://www.kitlender.com/catalogue/>
   2. Turo: <https://turo.com/us/en>
   3. REI: <https://www.rei.com/stores/rentals>
2. Proposed High Level Requirements
   1. Non-functional requirements
      1. <https://www.stackhawk.com/blog/10-web-application-security-threats-and-how-to-mitigate-them/>
3. Deployment Plan
   1. Database: Elephant SQL: <https://www.elephantsql.com/>
   2. Front end: Netlify: <https://www.netlify.com/>

# Glossary

(Any acronym used in the document should be explained here)

1. Renting User- RU
2. Hosting User - HU
3. Structured Query Language - SQL