Hog Heaven

Billy, Jason,

Luke, Hog

### Table of Contents

[Table of Contents](#_7vbf9ja7xyoi)

[Mission Statement](#_j2qiseiwi40z)

[Database](#_xo49dqo7h1vn)

[Features](#_pamvuvex3ump)

[Trello Link](#_fcmgtowcu1fr)

[Wireframe](#_q85md16hzpog)

[Schedule](#_ye3wz1kg9ovv)

[Final Notes](#_9561h8mlbskk)

### 

### 

### Mission Statement

This is where your team writes a short description on what your project is, the target audience, and the reasoning behind your choice of your project.

### *An application for motorcycle enthusiasts to buy, sell motorcycles, chat with other enthusiasts. There is a lack of dedicated resources to the motorcycle scene, this would help people find events, and find a repair shop.*

### Database

This is where your team writes out how your database will look like. List out each table, the columns (include the dataTypes), and the database associations your project will have on the server-side.

Table 1

Users, use identity user class?

Name, user ID, bio, profile picture, your bike(s), preferred shop

Table 2

Repair shops, shop ID, shop name, part inventory, location, foreign key with parts, hours of operation

Table 3

Parts, part name and then part type, partID, model number, manufacturer

// Enum: engine, frame, wheel, tank, seat, exhaust, drive train

Optional:

Additional Tables

Finding events

Chatting

Endpoints

Decide in the next couple of days if we make our own location class, will we use google? Use a third party api for locations

### Features

Features are instances or examples of different pieces of functionality. This is where your team lists out the features you are planning on implementing. Consider the different steps and logic those features require to do the expected job. This could include fetching data from a 3rd party API or simply looping over data from your server. Differentiate between your version 1.0 or MVP (minimal viable product) and version 2.0 or stretch goals.

### 

|  |  |
| --- | --- |
| Version 1.0 / MVP | Version 2.0 / Stretch Goals |
| * User class * Repair shops * Database of parts | * Transaction functionality * Chat features * Find events |

### 

### Trello Link

Once you have your features listed out, start writing out the tickets that each feature includes on your team’s Trello Board. Paste the link here.

### 

### Schedule

This project is the equivalent of one sprint in the agile methodology. In this section, write out a schedule spanning over the next couple of weeks. This should include deployment, time set aside to tackle especially challenging features, testing, etc. Consider whether your team will be working over the weekend(s). The table below is a guideline. It is not necessary to specify each day’s work/logic. Feel free to estimate your time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Day 1** | **Day 2** | **Day 3** | **Day 4** | **Day 5** | **Day 6** | **Day 7** |
| Build scaffold, research the code necessary | Build out CRUDs | Build out CRUDs | Build out CRUDs | Build out CRUDs | Build out CRUDs | Have classes, CRUD all coded, built |
| **Day 8** | **Day 9** | **Day 10** | **Day 11** | **Day 12** | **Day 13** | **Day 14** |
| Merge all branches, bug fix and test | Test endpoints | Test functionality | Test endpoints | Bug fix | Make it good | Have everything fully tested, and functional |
| **Day 15** | **Day 16** | **Day 17** | **Day 18** |  |  |  |
| Polish code further or add stretch goals | Polish code further or add stretch goals | Polish code further or add stretch goals | Turn in project |  |  |  |

### Final Notes

Great job with planning! You are now set to start coding. Planning a project is incredibly beneficial to the success of your team and your project. Here are some resources to help you with your planning.

* [How to plan a web application](https://selftaughtcoders.com/plan-web-application/)
* [Step By Step: Planning a web application](https://medium.com/@ericwindmill/step-by-step-planning-a-web-application-ddaa010a8353)