horizontal line

Grand Traffic Auto

Project Plan

14th April 2019

# 

# 

# 

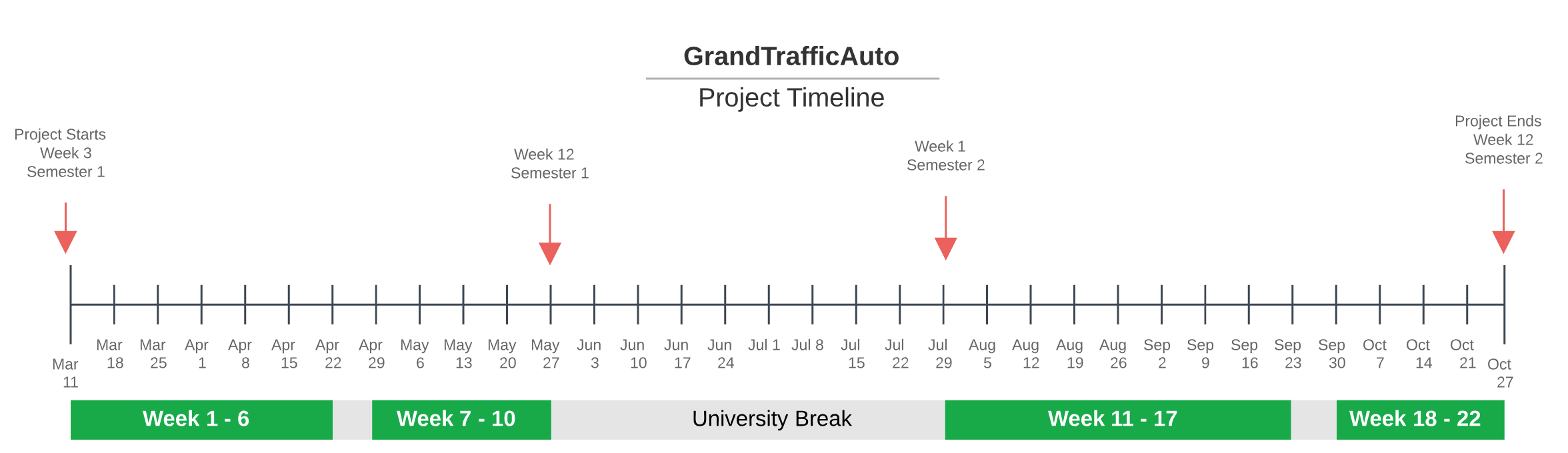
# **Project Team**

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| Robert Pennefather | 21511164 |
| Daniel Gonzalez | 22024722 |
| Martin Porebski | 21498791 |
| Jacqueline Soon | 21719676 |

## 

## 1.0 Project Timeline

Project work commenced in Week 3 of Semester 1, 2019 and is expected to finish in Week 12 of Semester 2, 2019. At minimum, members are committed to working during the running of university semesters, excluding university holidays and breaks. Therefore, the project will span approximately 22 weeks in total.



## 2.0 Commitment and Work Load

There are four members of the project team. Policy at the University of Western Australia states that 150 hours of workload is expected per student enrolled in a 6 credit point unit [[1]](http://www.governance.uwa.edu.au/procedures/policies/policies-and-procedures?method=document&id=UP17%2F3). Taking into consideration other workload commitments within the unit, members have agreed to contribute 200 hours each to the project, that is 9 hours per week. This includes all project related tasks including research, skills development, actual project work, team meeting and client meetings.

An approximate work breakdown per person for the entire 22 week project can be seen in the table below. Reasoning for each work estimate is also included below.

|  |  |
| --- | --- |
| **Project Commitment** | **Estimated Hours per person** |
| Team Meetings | 44 |
| Client Meetings | 10 |
| Project Documentation | 10 |
| Software Requirements Stage | 30 |
| Software Development & Testing | 106 |
| **Total commitment** | **200** |

**Reasoning for work allocation:**

Of particular note, members contribute working hours to the following:

* **Client Meetings:** Meetings will be weekly for the first five weeks of requirements analysis, then periodically for the remainder of the project as required. It’s estimated that there will be 10 client meetings throughout the project lifetime.   
  *Therefore, 10 x 1 = 10 hours per person*
* **Team Meetings:** The team will meet every week for 2 hours each week on average. *Therefore, 2 x 22 = 44 hours per person*
* **Software Requirements Stage:** this period lasted for 5 weeks. Weekly commitment is 9 hours, therefore excluding client and team meetings requirements took up 6 hours a week.   
  *Therefore, 6 hours x 5 weeks = 30 hours per person*
* **Documentation:** Team has allocated 10 hours per person for formal documentation
* **Software Development & Testing:** Deducting from a total work commitment of 200 hours, this leaves an estimated 106 hours for software development & Testing.

Each member is expected to contribute around 106 hours each to software development and testing which will run for 17 of the 22 weeks in the project. This is approximately **6 hours of work a week per person**. It should be noted that this is only a rough guide used for planning purposes. Realistically we expect that commitment hours could change week by week depending on each members availability throughout the year. Work progress will be monitored by the team and the project plan should be adapted if progress isn’t as predicted.

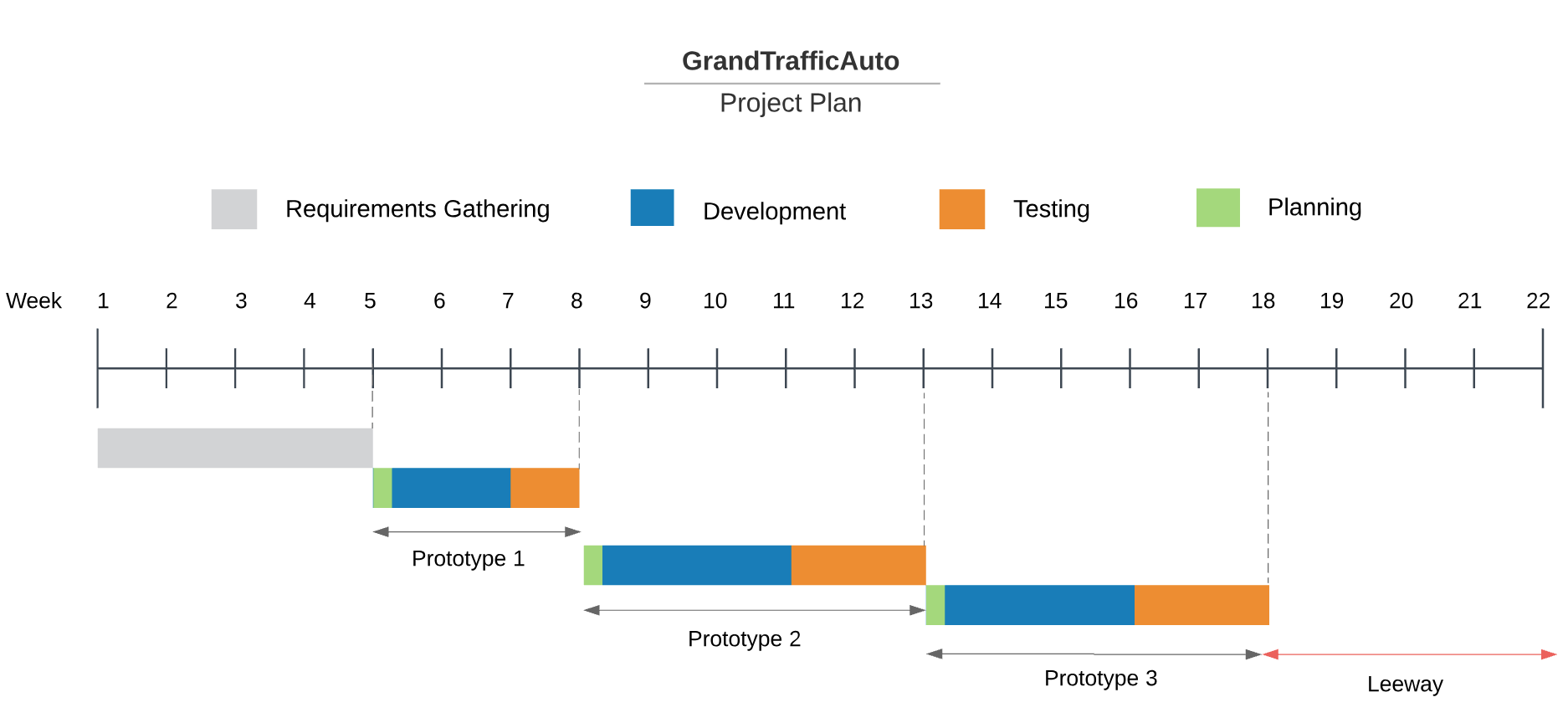
In total that is **424 work hours available for software development and testing** (106 x 4). This estimate has been taken into consideration for planning total work hours in section 4.0 Task Breakdown.

## 3.0 Project Management

The project will be managed based of an agile approach. Throughout the duration of the project there will be three sprints for Prototype 1, 2 and 3. Work during each prototype consists of:

* 2-3 weeks development,
* 1-2 weeks of testing, and
* 1 project planning session.

The purpose of the project planning session at the end of each prototype is to realign the project plan to the team’s current progress. This is an opportunity add or reduce the number of tasks for the next sprint. An overview of the project plan can be seen below:



The time span of each prototype is based of its estimated total work hours, more detail is included in section 4.0 Task Breakdown. The project plan includes a 4 week leeway which acts as a buffer in case the project does not go as planned. The reasons for this leeway have been discussed in Section 6.0 Contingency Plan.

## 4.0 Task Breakdown

Development tasks have been sorted into three distinct prototypes to align with prioritisation of requirements and the iterative nature of the project.

* Prototype 1, establishes the basic framework of the game
* Prototype 2, develops the architecture and mechanics of the game from start to finish
* Prototype 3, adds to the game logic providing more advanced user experience

Project work has been broken down into individual software development and testing tasks. Tasks are displayed as numbered cards where D# is a development task and T# is a testing task. The number in brackets shows the estimated working hours required for one person to complete the card.

|  |
| --- |
| D1. Create game session, single player and offline (Hours Allocated, Person Allocated) |

Example of task card

One of the goals of the project is for all members to develop a wide range of skills including software development, software testing, documentation and project management. Throughout the project members will be responsible for picking up 5-6 hours of tasks each week and completing them by the due date.   
  
Task have been allocated to each person based off accumulating equal work hours per person. In some cases, tasks focusing on one area have been allocated to the same person e.g. Adding both rigid body and drive functionality to car assets.

Due to the unpredictability of software development we have dedicated ample time in the case of time and scope creep for the project. Any additional time the team has once the main requirements of the project is complete will be dedicated to adding documentation and improving the readability of our code, such that the next team that works on the project can build on the current work.

The tables below show the development and testing task breakdowns for all three prototypes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prototype 1** | | | | |
| **Development**  Total Hours: 45  Approximate  Weeks: 2 | D1. Create game session, single player and offline (1, Daniel) | D5. Add rigidbody to landscape and car asset (2, Robert) | D9. Add collision prevention into car assets (3, Martin) | D13. Develop users ability to pan vision 180 degrees horizontally with mouse (3, Jacquie) |
| D2. Add car asset to the game (2, Robert ) | D6. Build drive functionality into car asset (3, Robert) | D10. Program invisible barrier to prevent user from driving off roads (3, Daniel) | D14. Set up testing framework (4, Daniel) |
| D3. Purchase unity assets for roads, buildings and traffic lights (4, Martin) | D7. Fix player POV to driver seat (1, Jacquie) | D11. Add rotation animation to car steering wheel (3, Jacquie) | D15. Add realistic graphic effects to assets (4, Martin) |
| D4. Add landscape assets to the virtual world (roads, buildings and traffic lights) (6, Jacquie) | D8. Build collision detection into car asset (3, Martin) | D12. Add rotation animation to car wheels (3, Robert) |  |
| **Testing**  Total Hours: 19  Approximate  Weeks: 1 | T1. Car has forward, left, right and reverse movement when selected keyboard input is detected (2, Robert) | T3. Player is unable to drive off roads (3, Martin) | T5. Car steering wheel turns appropriately with car (2, Robert) | T7. When player mouse moves, vision pans accordingly. (180 degree range) (2, Jacquie) |
| T2. Collision detection prevents car from colliding with other assets (3, Martin) | T4. When car moves, wheels rotate (2, Daniel) | T6. When car turns left, wheels also turn left and vice versa. (2, Robert) | T8. Player cannot drive off virtual world map (4, Daniel) |
| Total Weeks: 3 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prototype 2** | | | | |
| **Development**  Total Hours: 64  Approximate  Weeks: 3 | D16. Code login backend for Admin (4, Robert) | D20. Frontend screen for admin to upload JSON file to game. (3, Robert) | D24. Build frontend for Player to enter session code (3, Robert) | D28. Create timer for game session (4, Martin) |
| D17. Create frontend login screen for Admin (3, Jacquie) | D21. Create Admin ability to create sessions, make session code and end sessions (backend) (6, Daniel) | D25. Build frontend for Player to fill in survey (3, Jacquie) | D29. Add prompt for simple decisions during game (4, Daniel) |
| D18 Write mock configuration file (2, Daniel) | D22. Build frontend for admin to create and end sessions and view session code (4, Jacquie) | D26. Present form post-game for data on player experience (4, Robert) | D30. Add Survey at end of game (4, Jacquie) |
| D19. Create backend to read and store setting from csv, configuration files (6, Martin) | D23. Build backend for player to join session with valid session code (6, Daniel) | D27. Store survey data into database (3, Daniel) | D31. Randomly play 6 out of 12 decision sets (4, Jacquie) |
| **Testing**  Total Hours: 40  Approximate Weeks: 2 | T8. Test login details with screen (2, Robert) | T10. Test config file can be correctly uploaded (1, Robert) | T12. Front end screens are tested with client for usability, and adjusted (4, Jacquie) | T14. User acceptance tests for currently developed architecture, adjustments made where necessary (15, Everyone) |
| T9. Test config file is passing data correctly into the unity game (5, Martin) | T11. Session management is tested and works correctly (4, Daniel) | T13. Test survey information is taken in correctly and stored on to database (4, Daniel) | T15. Test for errors during incorrect front end input (5, Jacquie) |
| Total Weeks: 5 | | | | |

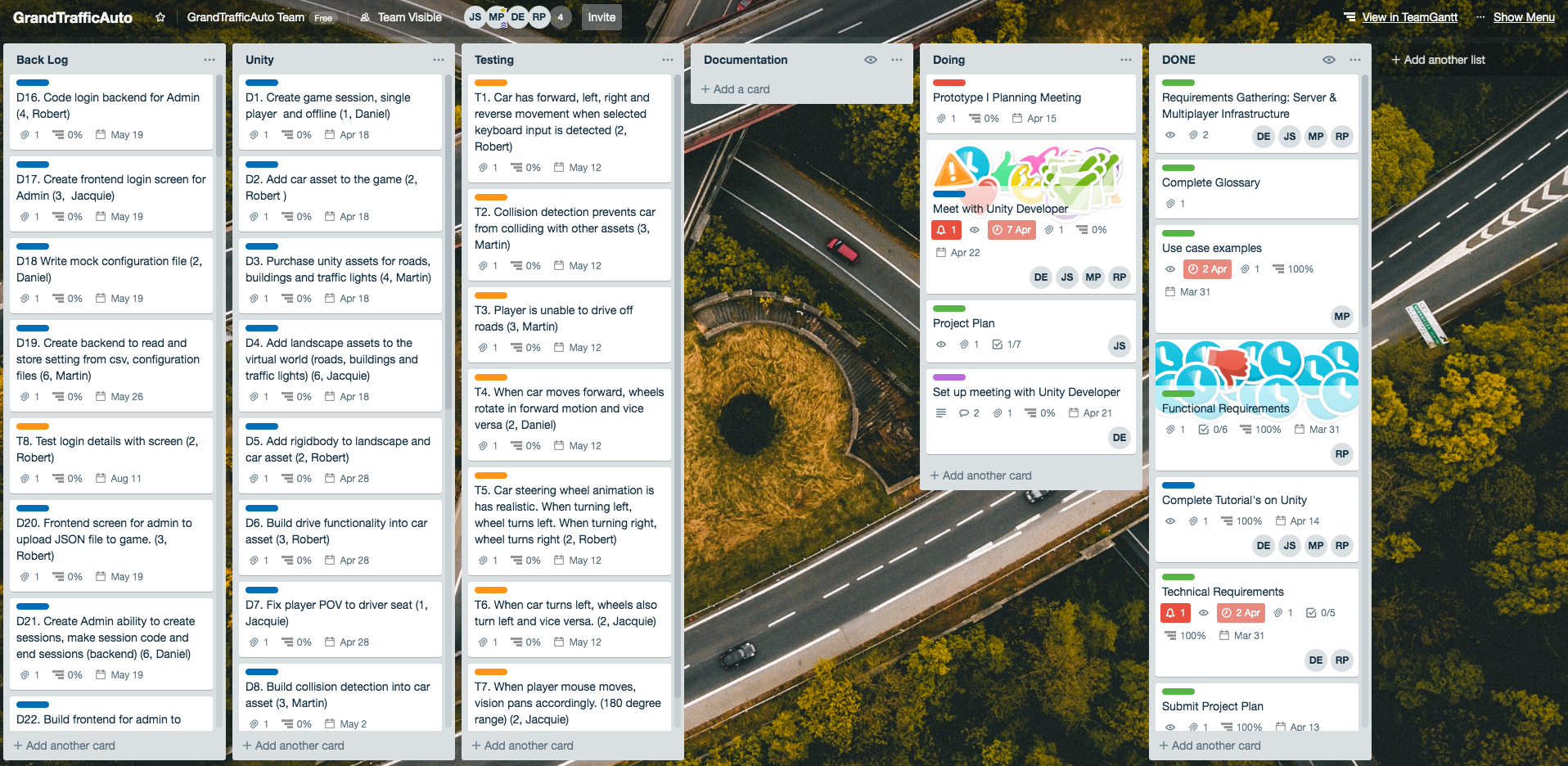
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prototype 3** | | | | |
| **Development**  Total Hours: 67  Approximate  Weeks: 3 | D32. Find and buy new civilian assets to the game (3, Daniel) | D37. Finalise assets for NPC cars and NPC civilians (3, Daniel) | D42. Create placeholder for survey, record data in database (3, Martin) | D47. Integrate csv file into game (6, Martin) |
| D33. Make civilians populate car when capacity changes (3, Daniel) | D38. Placeholder for player to select different route options while in AV (3, Daniel) | D43. Finalise and complete survey questions (2, Martin) | D48. Create game pause scene for walking (3, Daniel) |
| D34. Create AI for NPC car bots on road to populate streets (6, Robert) | D39. Make selected decisions have an effect on score (3, Daniel) | D44. Migrate game to local server (3, Martin) | D49. Integrate WebGL into game framework (5, Robert) |
| D35. Create logic for AVs to drive between any two points on map (4, Robert) | D40. Make selected decisions have an effect on route (3, Robert) | D45. Locate sound assets (3, Jacquie) | D50. Connect Survey database to WebGL (4, Jacquie) |
| D36. Create animation for player to initially get into the AV in first person (5, Jacquie) | D41. Record gameplay data in database (3, Martin) | D46. Add Sound assets to the game (4, Jacquie) | D51. Add passenger civilian assets to share ride vehicles (4, Martin) |
| **Testing**  Total Hours: 44  Approximate  Weeks: 2 | T15. Test to make sure capacity in car shows correct number of people (2, Daniel) | T18. Test the sound assets for correct timing (2, Robert) | T21. Test game and survey data is stored correctly in database (3, Martin) | T24. Test that CSV file returns errors if format is incorrect (4, Jacquie) |
| T16. Test performance gameplay of local survey (Martin, 2) | T19. Test selecting new route decision has appropriate effects on score and route (2, Robert) | T22. Test for unexpected user inputs in survey form (3, Daniel) | T25. Test Browser compatibility and performance for WebGL (4, Robert) |
| T17. Test logic of AV routing (3, Robert) | T20. Test different combinations of csv configurations (3, Jacquie) | T23. User acceptance tests for currently developed game, adjustments made where necessary (12, Everyone) | T26. Test that database connections through WebGL are working (4, Martin) |
| Total Weeks: 5 | | | | |

The completion of cards will be managed using Trello. Each of the task cards above has a reference number that will be used to correspond it to its Trello card, these will then be allocated to the appropriate person. The pace of project work can be monitored using TeamGantt, thereby allowing the team to make adjustments to the schedule when needed.

## 5.0 Gantt Chart

A current version of the project gantt chart can be found at the link below:

[Project Gantt Chart - TeamGantt](https://prod.teamgantt.com/gantt/schedule/?ids=1553398&public_keys=KwPsSgZOhiLo&zoom=d100&font_size=12&estimated_hours=0&assigned_resources=1&percent_complete=1&documents=0&comments=0&col_width=355&hide_header_tabs=0&menu_view=1&resource_filter=1&name_in_bar=0&name_next_to_bar=0&resource_names=1#user=&company=&custom=&date_filter=&hide_completed=false&color_filter=)

Gantt Chart tasks are automatically added and synced between Trello and TeamGantt. 

## 6.0 Contingency Plan

In total, the number of work hours to complete all development and testing tasks is 279 hours. The group have estimated 424 work hours, therefore there is a 145 hour leeway (Approximately 4 weeks). These hours will allow the team to have extra time to complete unforseen tasks.

Due to the nature of the project the client expects to continue developing the product into the future with other teams developing on our team’s work. Because of this, having a readable product with documentation makes it easily accessible for a new team. This is more important that completing all the specified requirements.

If requirements are not being completed at a rate as fast as the team had anticipated it would be beneficial to drop the requirement (given it’s not majorly important to a complete product) from the final product, and instead focusing that time on developing documentation, or improving the readability of code for the project completed so far. The team will develop the project using an agile approach, because of this iterative design process having a barebones product that’s useable should not be an issue for the team.

## 7.0 Continuous Integration

The team will develop two workflow methods for continuous integration, one for the website end of the project and a second for the game.

Work done on the website will have automatic tests run with every commit, confirming the success of previously developed tests. Once a team member is ready to merge their branch back into the master branch a second independent team member will manually read the code to check the readability and quality of code, before approving the merge into the master branch.

For unity each new major feature will be developed in a new git branch, a new set of tests will be developed alongside the new feature. Before approving a merge into the master branch the person developing the software will manually run the previously developed sets of unit tests, to confirm that nothing has been broken. After making sure the new code has not broken anything a pull request will be made into master. Another member of the team will read the code in the branch to check readability and quality of code, before allowing the merge into the master branch.

[1] http://www.governance.uwa.edu.au/procedures/policies/policies-and-procedures?method=document&id=UP17%2F3