**Final Report on the “Software Product” Groupwork**

*A taxi management system*

AlbaniaSuperUberApp Team: *Dalird Bufi, Melvir Singh, Xhulio Bishtaja*

1. Introduction

Project Description   
This project is a school project for the Applied Software Engineering module. We have named the project ‘AlbaniaSuperUberApp’. The motivation behind this product is 2 fold. Firstly, to be able to harness technology and be able to stamp our footprint on what we really want the current Uber / Taxi systems out there to be. By developing something from scratch, we are able to create features that we deem will be useful. Secondly, we would like to put into practice what we have learnt throughout the module. These include the programming technologies such as Flask and Python and also the project management aspects such as Scrum which we utilized to run the entire product from start to end.

The product was developed for a duration of about 1 month. Unfortunately, 1 month is never sufficient to deliver everything that we would like in an ‘Uber’ type of system but we focused on the core features and believe we have made it happen successfully via our 3 man development team.

We feel there are 3 aspects to the novelty of the product. Firstly, the product is based on a proper client/server based architecture. The benefit of this is that the server side can be started small (due to just being a 1 month development project work) with a handful of RESTful API services and be scaled up over time. Thus, the system can always be built-on and it will never be the end to it as the code base will remain manageable and sustainable. Secondly, we feel that by including all 3 user roles into the system, the product is self-sustaining. For example, we have the Customer, Driver and Provider roles. Each role comes with their own set of features and a user can simply sign up and be completely self-sustaining instead of requiring any form of manual intervention. Lastly, we have the Feedback feature in the product such that Customers and Drivers can provide feedback in real-time in the application that the Provider can take note of and potentially include in subsequent Sprints. We have to be upfront that there are no novel Customer features over the current Uber app in the market but it would be silly to expect such in a 1 month development timeframe. Moreover, in any proper product management lifecycle, core features should be prioritized and developed first prior to other unique/advanced/novel features.

Challenges   
There were 2 main challenges we encountered throughout the project.

Firstly, other than Melvir, the other 2 members of the team are completely new to developing in the Client/Server architecture. Moreover, the architecture was covered at the very most 1 hour in a lesson and there was no practical lessons that the team members could try their hands on to try it out. Rather, this project became the practical lesson. Is that ideal? I think it depends on what the outcome and deliverable of the product is. However, we got through this challenge by Melvir sharing his experience in developing in such an architecture, giving concrete code examples as reference that the rest could follow. We also followed this up by spending lots of time and effort following online tutorials to learn the basics to be able to code in proper standards.

Secondly, it’s also the first time that the 3 of us are working together on something. Naturally, there will be scenarios where we are learning/adapting to each other’s working styles. Hence, we went through the typical group forming, storming, norming and performing lifecycle. We kept things objective on the project knowing that we had certain deliverables to meet. The scrum model did help as it helped to set tasks in smaller attainable ways.

Highlights   
We are very pleased to report that we feel that the entire team has come to a good standard of being able to code well and produce features quickly in the Client/Server architecture by the end of the project. We also worked well in the Scrum model and all of us adapted to the Planning, Daily Scrum (over 3 days instead of daily) and Review meetings well and were very consistent throughout the project lifecycle. Furthermore, all 3 of us got to be good friends and will continue to network even after the end of this module.

Changes

* Summarise any significant changes to any aspect of the project during the module.
* Include the date, motivation, description, and implications of each change.

1. Customer Needs

We feel that with some finetuning, the product could potentially be trialed in a proper market with Users able to sign up as either Customers or Drivers. Both these roles form the primary customers in the market for us. We do not have a specific planned customer in mind as we developed the product as an overall fit to the market and believe that it could be successful as a competitor to Uber with the features that we have built (and can potentially finetuned further).

We did User Journeys with some Customers and Drivers and realized that some of the pain-points they brought up (such as not being able to submit real-time feedback in app or not being able to export their Booking/Job history) which we have implemented in the product would definitely be useful and engaging to convince them to try out the product.

In terms of secondary stakeholders, we believe that they would include Citizens of Albania that we’re launching the application in as well as the media in general. For the Citizens, it definitely allows them a different option to Speed Taxi App. The benefit of such is that should Uber start to raise prices unfairly, they would be able to compare the prices to our product and thus still have a viable option to continue booking taxis. For the media, it will be a 2 way opportunity for them and for us to be able to provide a review of the product. Should the product be successful, it might draw more views to their review and thus their media company overall as well.

The desired overall experience that we wish to accomplish for our stakeholders is a good, consistent, intuitive product for them to trial and use as an alternative to Speed Taxi App in Albania.

User Requirements (**Giulio** – quite straightforward with what we already have)

* Include SMART user stories written using the “As a ... I want ... so that ...” template.
* Include acceptance tests for the user stories using the “Given ... when .... then ...” template.

*Documentation* (**Giulio** – quite straightforward with what we already have)

Briefly describe the following:

* Personas,
* user stories,
* user stories, and then the resulting
* feature identification.

The detailed descriptions should appear in other documents.

1. Project Goals

Since Albania does not have an Uber like app, we wanted to deliver a product that could be a solid competitor to the taxis and their current Speed Taxi app there.

Thus, the main customer problem we chose to solve and that we wanted to improve the experience of was to ensure the entire booking process intuitive and smooth. For the customer, we also wanted them to be able to provide us and the providers real-time feedback.

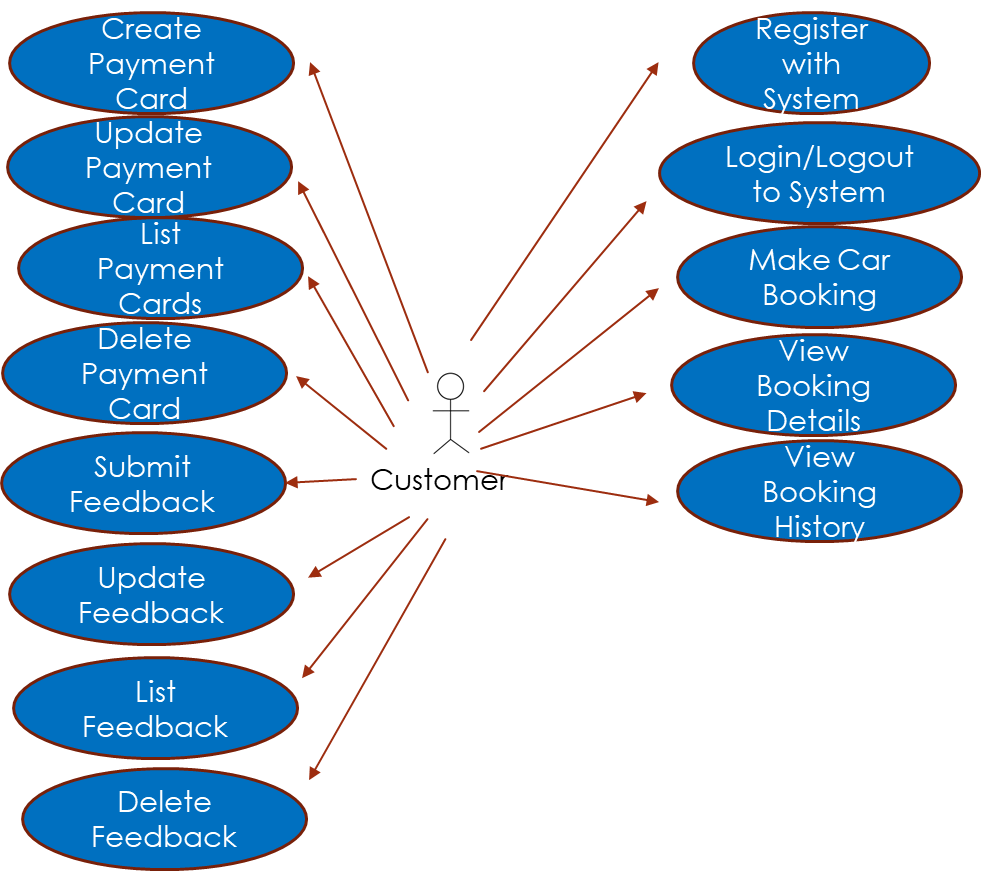
The main driver problem we wanted to solve was to allow them a comprehensive overview of all the jobs they have undertaken. Compared to the Speed Taxi app, the drivers feedback that the interface was buggy and often resulted in errors leading to them not being able to view their job history.

For the providers, the main problem we wanted to solve was that there was no current way to have a Command & Control view of all their cars in real-time. Thus, we wanted to achieve a C&C view for them to have a quick overview and sensing of all the cars and their respective locations.

We believe that by focusing on these 3 issues, we would be able to deliver a product that fit well for all 3 roles that we are releasing the product for – Customers, Drivers, Providers. The benefits would not only be in terms of time saved but also in terms of effort saved. We validated these ideas during the User Journeys with our users (covered above) and they were all very positive about the experience and benefits that our product will come with. That was indeed very encouraging.

Uses Cases

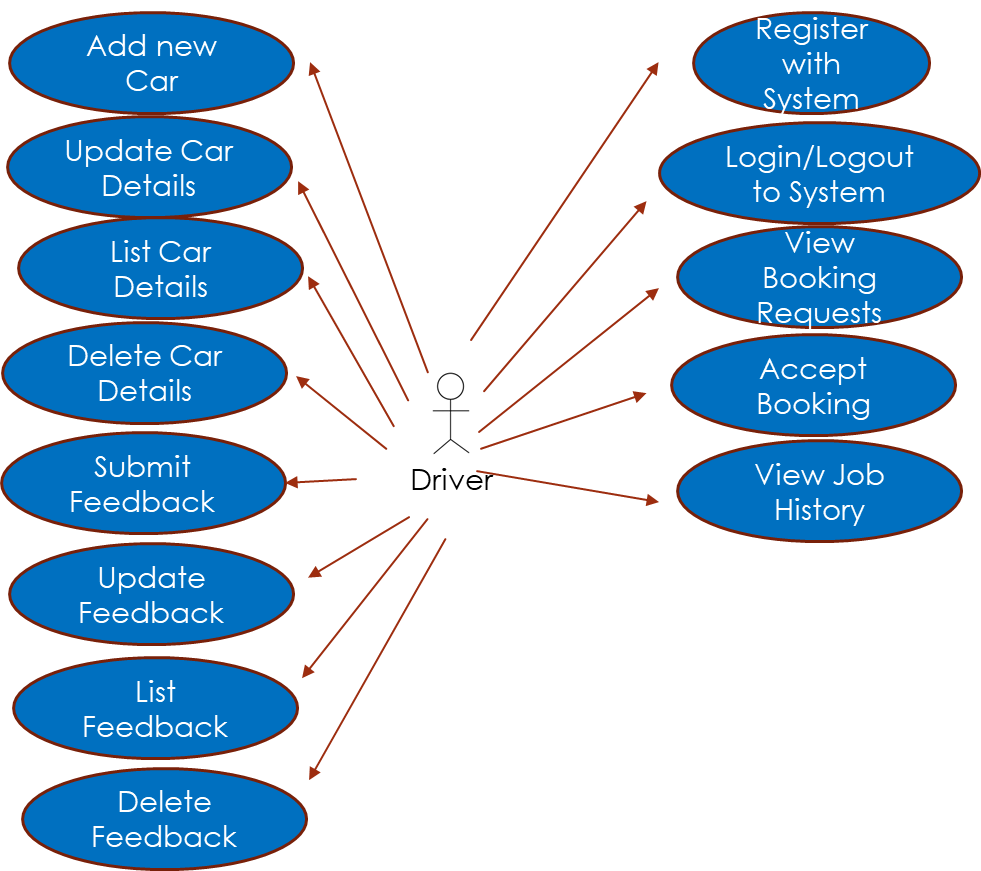
**For Customer Role**



**Detailed Description of 1 use-case for Customer Role**

|  |  |  |
| --- | --- | --- |
| Use-Case | Make Car Booking | |
| Description | Customer shall be able to make a booking for a Car to get from Current Location to Destination | |
| Precondition | The user has registered successfully with ‘Customer’ role and has also logged in successfully. | |
| Sequence | Step | Action |
| 1 | User logs into application |
| 2 | User clicks on ‘Book Car’ button on top navigation bar |
| 3 | User inputs in Current Location (eg. Wembley) and Destination (eg. Liverpool) |
| 4 | User clicks on ‘Book’ button |
| 5 | System updates user that the request has been sent and awaiting for acceptance by any Driver |
| 6 | User clicks on ‘Check if accepted by Driver (again)’. Assumption that a Driver has already accepted |
| 7 | System updates user with Car details |
| 8 | User clicks on ‘Simulate Starting of Ride’ |
| 9 | System updates user that Ride is started and updates Driver’s car pos X and pos Y with the Current Location specified |
| 10 | User clicks on ‘Simulate Completion of Ride’ |
| 11 | System updates user that Ride is completed and updates Driver’s car pos X and pos Y with the Destination specified |
| 12 | User selects Driver Rating and clicks on ‘Rate your Driver’ |
| 13 | System thanks user for the rating and informs the end of flow |
| Postcondition | Driver’s car’s pos X and pos Y are updated to the Destination longitude and latitude | |
| Comments | The steps are longer than a typical use-case on Uber as the current application is not able to automatically trigger starting and completion of ride and thus requires the User to push some buttons through the process | |

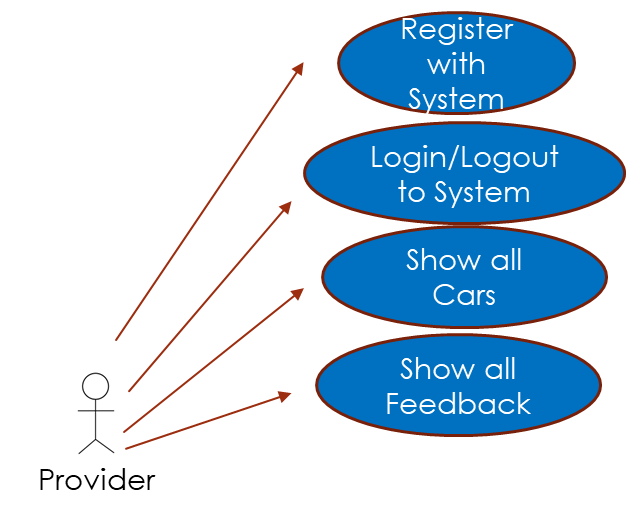
**For Driver Role**



**Detailed Description of 1 use-case for Driver Role**

|  |  |  |
| --- | --- | --- |
| Use-Case | Accept Booking | |
| Description | Driver shall be able to accept booking requests from Customers to get them from Current Location to Destination | |
| Precondition | The user has registered successfully with ‘Driver role and has also logged in successfully. | |
| Sequence | Step | Action |
| 1 | User logs into application |
| 2 | User clicks on ‘View Booking Requests’ button on top navigation bar |
| 3 | User looks through the Booking Requests that appear in a table format |
| 4 | User clicks on ‘Accept’ button for the ride that he/she wishes to accept |
| 5 | System updates user that the request has been accepted and that the Customer will be notified |
| 6 | System updates Booking table with the row’s ID to ‘Accepted by Driver’ status and car\_id to ID of Driver’s car |
| Postcondition | The accepted booking request’s row will be updated in the database with ‘Accepted by Driver’ status and the car\_id will also be populated | |
| Comments | The steps are straightforward and intentionally allows the Driver to freely select which job he/she wishes to accept rather than only showing those requests which are very close to him | |

**For Provider Role**



**Detailed Description of 1 use-case for Driver Role**

|  |  |  |
| --- | --- | --- |
| Use-Case | Show all Cars | |
| Description | Provider shall be able to show all the cars in their system into a nice overview of Map to have a Command & Control sort of view | |
| Precondition | The user has registered successfully with ‘Provider’ role and has also logged in successfully. | |
| Sequence | Step | Action |
| 1 | User logs into application |
| 2 | User clicks on ‘Show all Cars’ button on top navigation bar |
| 3 | Map loads with all the Cars in their correct positions |
| 4 | User can click on any of the Cars that appears on the map to see the car brand, model and ids |
| Postcondition | Page loads without any errors | |
| Comments | The steps are straightforward and allows the Provider to see an overview of all cars on the map | |

Measures of Success

We believe that measuring success of our product and its use-cases are critical. Thus, we plan on doing this via User Interviews and also Surveys. For the former, we plan on meeting up with the same users that we did the User Interviews with to check in with them after they have trialed the app. This will allow them to provide an honest assessment on whether their pain points discussed before had indeed been addressed by our product. For the latter. We plan on coming up with a Survey questionnaire asking users on aspects such as benefits, functionalities, intuitiveness, UI/UX. We would then compile the results and assess how successful our product has been. We would also be able to prioritize their feedback by creating PBIs to add on to our Product Backlog.

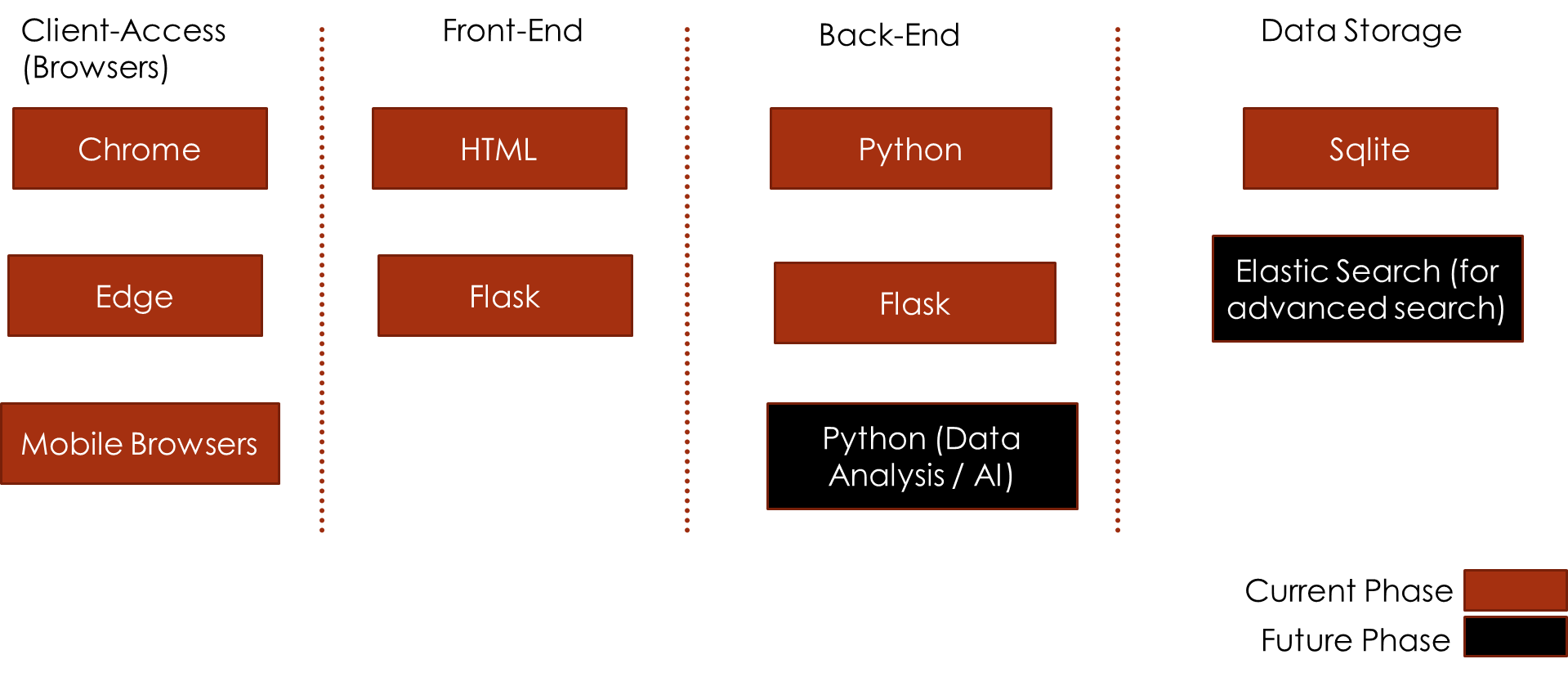
1. System Description

System Overview

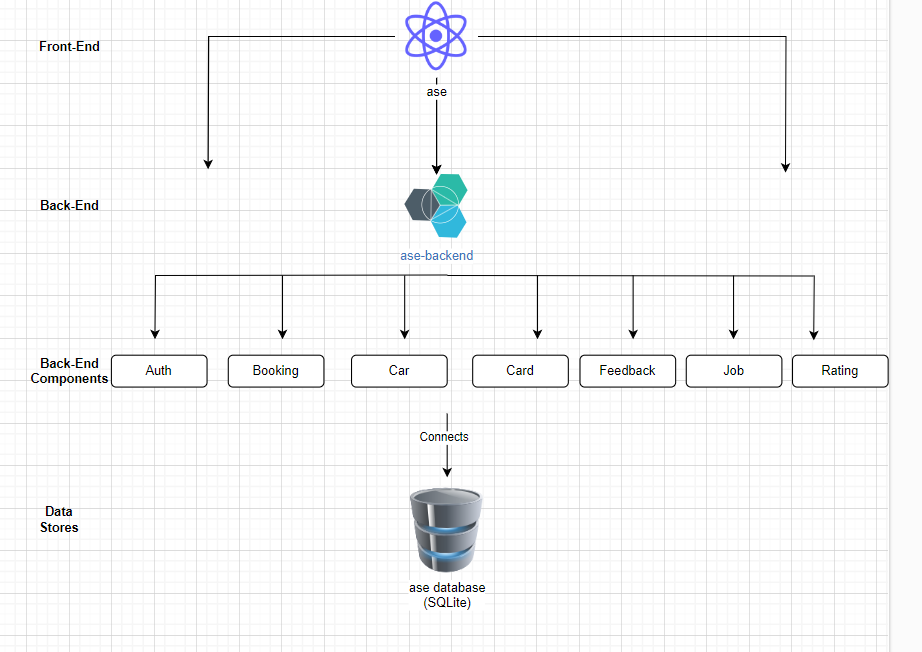
The product uses the Client/Server architecture, thus having separate frontend and backend applications. There are various benefits to doing a product in such an architecture. Firstly, it allows newcomers to be able to understand each part easier and quicker as the codebase is naturally smaller due to it being split. Secondly, deployment is also more straightforward as any changes in frontend will only require re-deployment of the frontend without affecting the backend. Lastly, there are proper segregated concerns of responsibilities for having them separately. For example, the frontend is purely responsible for handling client (user) requests before sending the payloads to the backend. The backend is purely responsible for business logic and to perform data manipulation to/from the database.

Naturally, there are challenges to this architecture too. Firstly, it is definitely more complex and complicated to code in compared to having both frontend/backend together in a single application. The technologies are also different (one contains flask/html whilst the other contains flask/python) and thus developers would need a greater range of skillsets. Secondly, it can also be harder to deploy as the team needs to ensure both deployments are running properly compared to just having to monitor one if both were packaged together.  
  
However, the team feels that the benefits greatly outweighs the drawbacks thus we decided to proceed with this architecture. Should the team pick up more knowledge on tech standards in future the backend could be further decoupled/decomposed into several different backends to take advantage of the benefits that microservices entail as well.

Technology Stack

****

Application Architecture



There is a single frontend application that interacts with a single backend application. However, the backend application serves API routes on various different components. Each of this components take care of their own set domain boundary. For example, all authentication (such as register/login/logout) APIs are stored in the auth.py file. It makes it self-explanatory and also aids troubleshooting allowing faults to be narrowed much quicker. If there are any new authentication data required from the database, a new API route can be created in the auth.py file which will then be served by the ase-backend service.

Below is a summary of the API routes created for each backend component and also the team member that was in-charge to lead the development for it. Naturally, the same person who worked on that backend component also worked on the same frontend component to take care of the display on the pages. However, in light of having consistent code, we came up with proper code standards and flow. Then, with that as reference code and design flow, all of us managed to code well with proper standards. Thus, a new developer joining would not get a shock encountering completely different code and flow from page to page.

Components Breakdown & Member in-charge

|  |  |  |  |
| --- | --- | --- | --- |
| Back-end Component | API routes created | Description of their usage | Team Member in-charge |
| Auth | /api/register /api/login  /api/<int:id>/loadUser | For user registration For user login To load user information into session upon successful login | Melvir |
| Booking | /api/showCars  /api/bookcar  /api/listRequests  /api/checkBooking  /api/<int:id>/acceptJob  /api/<int:id>/getCarId  /api/<int:id>/getCarDetails  /api/listBookings  /api/startBooking  /api/completeBooking  /api/rateDriver | To display cars for Provider To start car booking process for Customer To list all requests for Driver to accept To check if booking has been accepted by any Driver For Driver to accept job  To retrieve car ID To retrieve car details For Driver to show jobs history To simulate ride start once Driver arrived at location To simulate ride completion once Driver arrived at destination  To rate driver | Irdi |
| Car | /api/listCarDetails  /api/createCar  /api/<int:id>/getCar  /api/<int:id>/updateCar  /api/<int:id>/deleteCar | To list car details To add new car for new Driver To get car from car ID To update car details To delete car for Driver | Giulio |
| Card | /api/listCard  /api/createCard  /api/<int:id>/getCard  /api/<int:id>/updateCard  /api/<int:id>/deleteCard | To list payment card details To add new payment card for Customer To get payment card from card ID To update payment card To delete payment card for Customer | Melvir |
| Feedback | /api/listFeedback  /api/listAllFeedback  /api/createFeedback  /api/<int:id>/getFeedback  /api/<int:id>/updateFeedback  /api/<int:id>/deleteFeedback | To list all feedback details To list all Feedback for Providers  To create new feedback  To get feedback from feedback ID To update feedback To delete feedback | Melvir |
| Job | /api/listJob | To list all booking job history for Drivers | Irdi |
| Rating | /api/driverRating | To get rating for Drivers | Giulio |

1. Final Status (Giulio – Show all the features/screenshots by each role by role. Need to make sure this part is really detailed. For test just say Unit Testing and also our manual Testing via the UI)

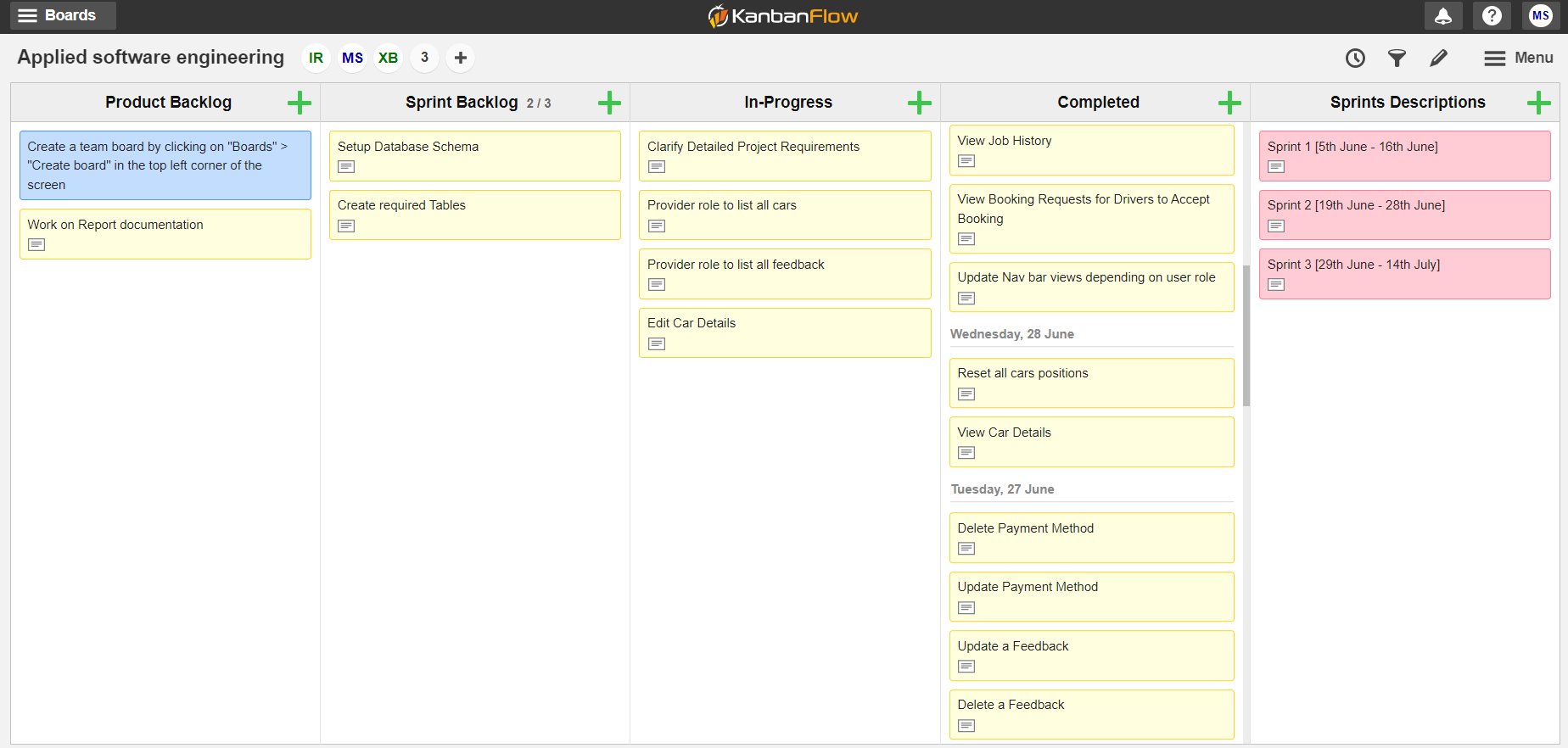
* What works? Include screenshots.
* What tests have you run?
* Estimate the adequacy of your tests.
* How many lines of code has the team written altogether?
* Did you use any existing codebases?
* What existing systems did you use as motivation?

1. Project Management

Software Methodology  
To run the entire project end to end, we depended on the industry standard and trustworthy agile methodology. We utilized the Scrum approach for the entire project. Some of us in the team have had some experience leading/being involved in a Scrum approach for projects before so it became much easier to set it as a standard. Moreover, it was also taught intensively in a previous module called Information Systems. Our “Kanban Board” was setup in an online tool called “Kanban Flow”. Prior to selecting that, we researched and trialed out various tools including one recommended by Keith called “Trello”. However, we found the latter an overkill as we already had MS Teams set up for the standard meetings and a WhatsApp group for informal communication. We were very cautious into finding simple and streamlined tools that made the entire project management conducive and efficient, rather than an overload of tools that incorporated buzzwords (less is more mantra).

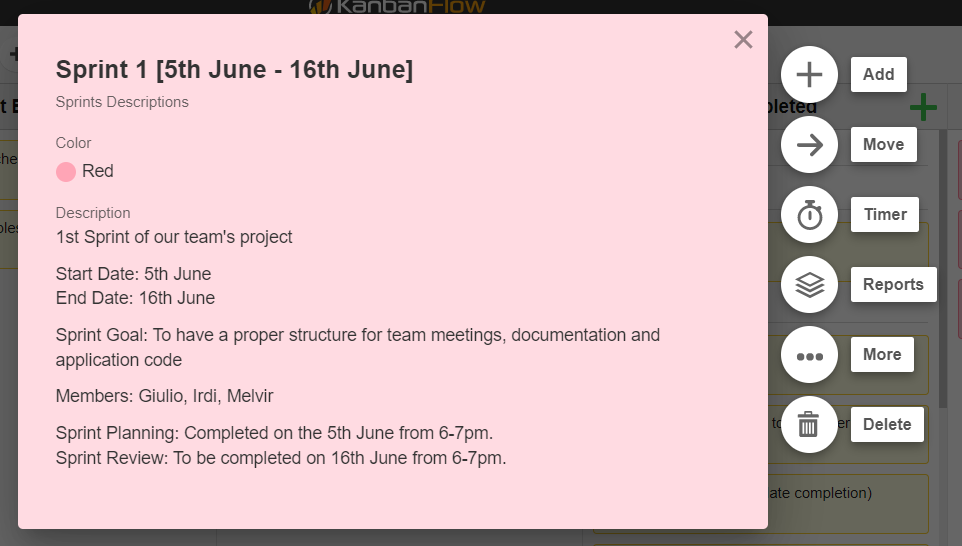
Tech Tools Utilized

|  |  |
| --- | --- |
| Use-Case | Tech Tool |
| To Host/Aid Meetings Online | MS Teams |
| To be our Kanban Board to hold Product Backlog & More | Kanban Flow Our team’s board link:  [Applied software engineering - KanbanFlow](https://kanbanflow.com/board/L6rwAHJ) |
| To Host our shared repositories | Frontend repository (it is public):  [melvir86/ase: This is a shared code repository for Group 3's Applied Software Engineering (ASE) project (github.com)](https://github.com/melvir86/ase)  Backend repository (it is public):  [melvir86/ase-backend: Backend for our ASE Ride Hailing project (github.com)](https://github.com/melvir86/ase-backend) |
| To be our individual application coding/testing platforms | Codio (Melvir):  [Codio - Teach Code. With confidence.](https://codio.co.uk/msingh3/the-product/)  Codio (Giulio):  [Codio - Teach Code. With confidence.](https://codio.co.uk/msingh3/the-product/) (Edit this link with your Codio info)  Codio (Irdi):  [Codio - Teach Code. With confidence.](https://codio.co.uk/msingh3/the-product/) (Edit this link with your Codio info) |

Significant Events  
Naturally the most significant events encompass around the 3 Sprints we ran from start to end. We have copied the material here from our Kanban Board. Prior to explaining the 3 Sprints, we have attached a screenshot showing details of our Kanban Board.  
  


Sprint 1

Description



Product Backlog Items (PBIs) worked on

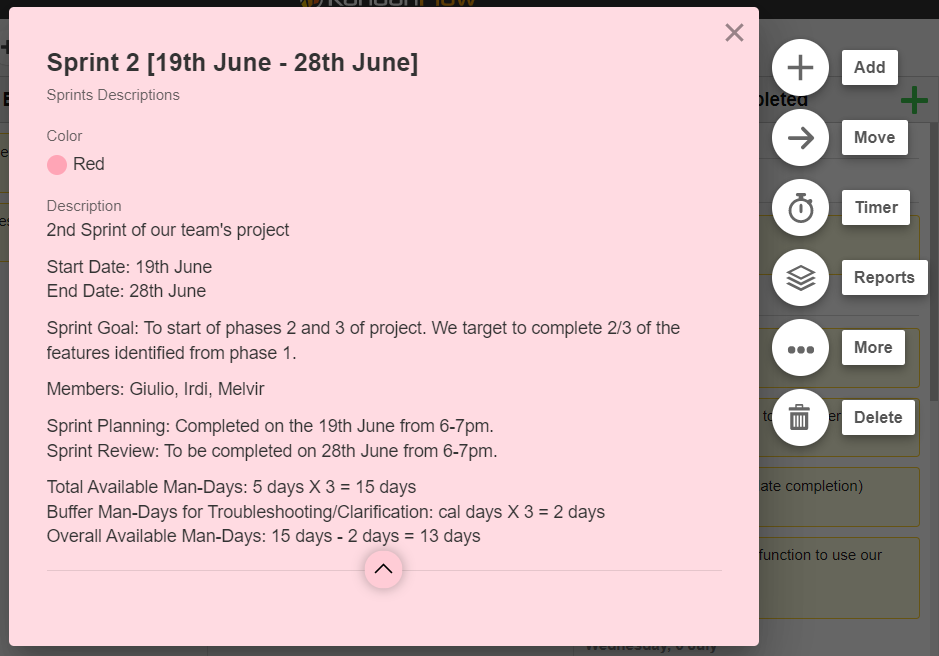
|  |  |
| --- | --- |
| Number | Item |
| 1 | Setup shared code repository |
| 2 | Research Shared platform for Product Management |
| 3 | Come up with Project Documentation Outline |
| 4 | Come up with Documentation for Scenarios |
| 5 | Come up with Documentation for Personas |
| 6 | Create Car class to store provided sample data to persist in database |

Major meetings/discussions

|  |  |  |
| --- | --- | --- |
| Description | Date | Recording (if any) |
| 1. Sprint 1 Planning | 9th June 2023 | Unfortunately the recording had an issue, thus we could not get a link from MS Teams |
| 1. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days] | 12th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/EV7tJSKXIOBNtejz4o7kxQ4BqwOXvG3OCZmy-zX9KpEXJw |
| 1. 3. Sprint 1 Review | 16th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/Ee2avMIPjWtMj0lwl8ZOANIBk41WuE9U8WauDh\_QjFqohw |

Sprint 2

Description



Product Backlog Items (PBIs) worked on

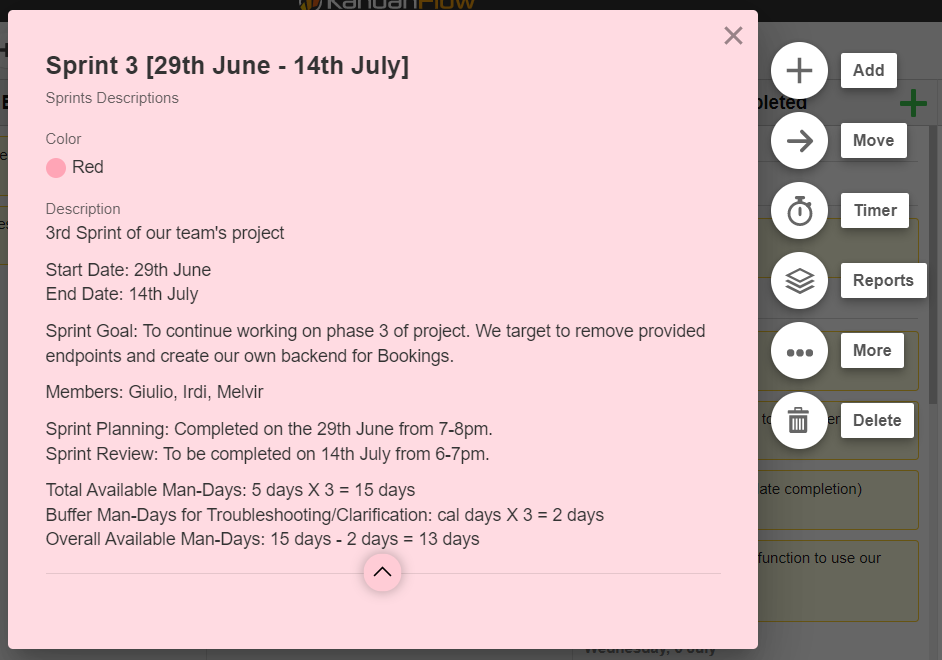
|  |  |
| --- | --- |
| Number | Item |
| 1 | Come up with Documentation for User Stories |
| 2 | Come up with Documentation for Feature Identification |
| 3 | User Registration |
| 4 | User Login |
| 5 | Display Car positions on Map |
| 6 | Make a Booking |
| 7 | Cancel a Booking |
| 8 | List all Payment Methods |
| 9 | Create new Payment Method |
| 10 | Update Payment Method |
| 11 | Delete Payment Method |
| 12 | Reset all Cars Positions |

Major meetings/discussions

|  |  |  |
| --- | --- | --- |
| Description | Date | Recording (if any) |
| 1. Sprint 2 Planning | 19th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/ETlqbuultI5FgM8alj8Y2IQBSQn2GyZOvNlmKImP9LLysA |
| 1. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days] | 24th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/ESE5g1Q7oLpNufML8KgzREIBurCeOSEoWjYhQUpgAxOImQ |
| 1. Daily Standup Simulation [what we did over the last 3 days and what we plan on working on for the next 3 days] | 28th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/EVVg\_RNaWaFHkeuktROAtScBFoOcpyikgQtSK4INfj5gAA |
| 1. Sprint 2 Review | 28th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/ETKz\_BMB1bBGvWTU75NzIw8BA1kXU5yA7x1Ui28cgMursw |

Sprint 3

Description



Product Backlog Items (PBIs) worked on

|  |  |
| --- | --- |
| Number | Item |
| 1 | Clarify Detailed Project Requirements |
| 2 | Simulate Car movement to Customer |
| 3 | Change architecture from monolith to Client & Server architecture |
| 4 | Update Book Car function to use our own logic |
| 5 | Design Database Schema |
| 6 | Create Required Tables |
| 7 | View Booking Requests for Drivers to Accept Booking |
| 8 | Simulate Button (to simulate Trip Completion) |
| 9 | Submit Driver Rating |
| 10 | View Driver Rating |
| 11 | View Job History (for Drivers) |
| 12 | View Booking History (for Customers) |
| 13 | Update Navigation bar views depending on user role upon login |
| 14 | Provider role to list all cars |
| 15 | Provider role to list all feedback |

Major meetings/discussions

|  |  |  |
| --- | --- | --- |
| Description | Date | Recording (if any) |
| Sprint 3 Planning | 28th June 2023 | https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/ETI7cT8uJhxGosLitJk7SKYBtEqinVwhBHFus66nj1H7fA |
| Daily Standup Simulation [what we did over the last 7 days and what we plan on working on for the next 7 days] | 5th July 2023 | (Part 1) - <https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09_student_bbk_ac_uk/EfCCz7Hq9AJJoyiEUM9P068BzNNwQKyBca1s6FFODKoBJw>  (Part 2) - https://birkbeckuol-my.sharepoint.com/:v:/g/personal/msingh09\_student\_bbk\_ac\_uk/Efc0fF\_PuOFCnwBNekLPkw0BlRYk-rHUjSU9TAT7HjhB7A |
| Daily Standup Simulation [what we did over the last 7 days and what we plan on working on for the next 7 days] | 12th July 2023 | TBC |
| Sprint 3 Review | 14th July 2023 | TBC |

Code Structure (Github repository that we pulled for our Codio platform)

We have 2 applications to serve the product – ase (frontend) & ase-backend (backend). There are multiple branches for each of this application that is described in the table below.

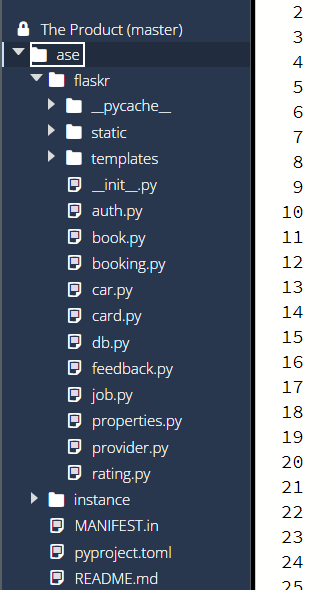
|  |  |  |
| --- | --- | --- |
| Application | Branches | Rationale |
| ase | main | Holds documentation folder (scenarios, stories, personas, feature-identifications, tech diagrams, project report) and meeting recording links |
| phase-one | Holds project code for phase-one functionalities and README file (to specify how to start the application) |
| phase-two | Holds project code for phase-two functionalities and README file (to specify how to start the application) |
| phase-three | Holds project code for phase- three functionalities and README file (to specify how to start the application) |
| ase-backend | main | Nil |
| phase-three | Holds project code for phase-three backend functionalities and README file (to specify how to start the application) |

Project Structure

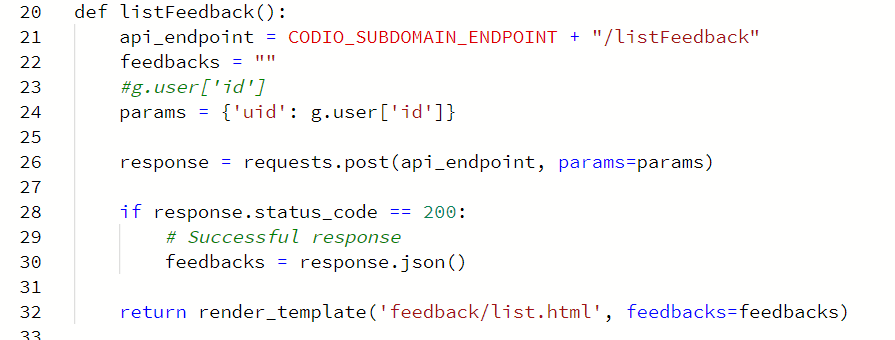
Frontend

For the frontend, our team decided to have just 2 subfolders (static & templates). The former holds the CSS file while the latter holds the various HTML templates to be served to the client (user).

We also have individual component py files that serve as the middleman between the frontend and backend applications. These py files (1) retrieves information from the HTML templates and calls the backend APIs with the required payloads & (2) sends the information to be loaded to the HTML templates using the JSON results returned from the backend APIs.



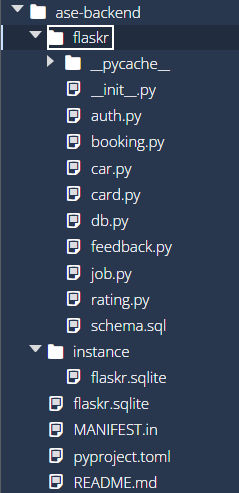
Below is an example of the Feedback component doing what was described above. In this scenario, the Feeback components calls the backend API endpoint and returns the resulting JSON to the Feedback template (feeback/list.html).



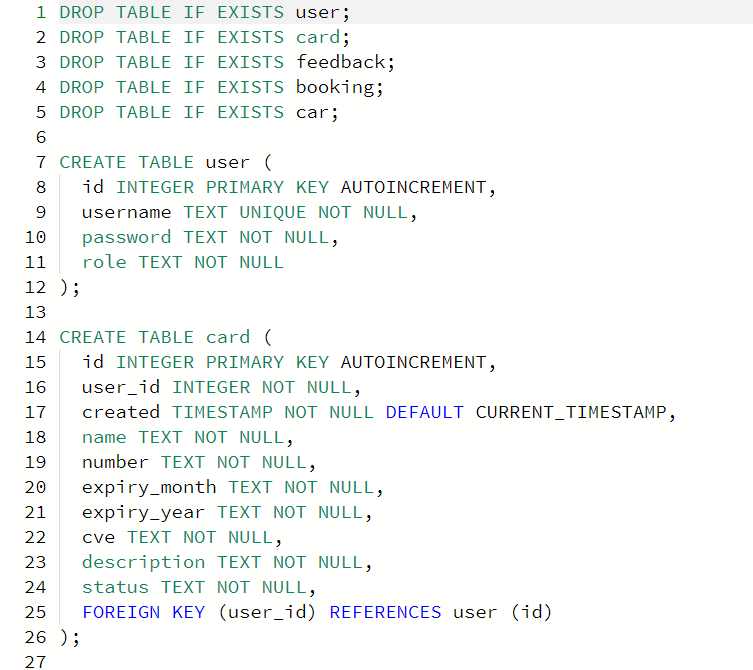
Backend

For the backend, it is more straightforward since its responsibility is only to serve API routes. Thus we just segregated the API files (py files) according to their different business domains (ie. car.py, feedback.py, etc).

Since the backend requires a database (SQLite) to store/retrieve data from, we have sqlite file that is stored in instance/flaskr.sqlite when initialized (flask --app flaskr init-db).



We also have a schema.sql that contains all the tables creation as well as for sample data to be added as shown in the screenshot, which allows us to test the application easily.





Lastly, we also have a single db.py that initializes the DB using the schema.sql file. This file also allows the various backend component python files to retrieve a DB connection easily, and in a standardized manner. This avoids us having to duplicate code everywhere, resulting in good code quality and reusability.

**

1. Team

Backgrounds

As shown in the table below, the only person with experience / expertise of building such applications/products was Melvir. Thus, it was a very uphill task from start of project to the end for the entire team. The team had to put up many late nights so that we were able to deliver a good product with solid code standards on time.

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Experience of IT / Tech | Experience building such a product | Experience with Scrum & Scrum tools |
| Melvir Singh | 5 years as Software Engineer, 3 years in DevOps and 2 years as Product Manager | Vast experience in building frontend and backend applications with specialization of containerized microservices | Vast experience in leading teams with Scrum methodology |
| Dalird Bufi | 3 years as Infrastructure Engineer | Nil | Nil |
| Xhulio Bishtaja | Completely new to IT / Tech | Nil | Nil |

We strongly believe that the team is a very well-functioning one, and one that allowed all team members to put in their fair share of work. With Melvir leading the team, the rest were able to pick up on the code standards and come up with features completely on their own later on. This further reiterates the importance of spending a fair bit of time laying good foundation that everyone in the team can appreciate and learn from.

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role | Features Contributed | Rough Percentage Contributions |
| Melvir Singh | Lead Architect / Developer | Customer – List all Payment Cards Customer – Submitting new Payment Card Customer – Updating a Payment Card Customer – Deleting a Payment Card Customer – View Booking History Driver – View Job History Driver – Accept Booking Requests Final Report | 33.33% |
| Dalird Bufi | Core Developer | Customer – Booking Car Customer – Simulate Starting of Ride  Customer – Simulate Completion of Ride Provider – Show all Cars on Map Unit Testing on all Backend components Creating Test/Sample Data for Testing Final Report | 33.33% |
| Xhulio Bishtaja | Core Developer | Customer – List all Feedback Customer – Submitting new Feedback Customer – Updating a Feedback Customer – Deleting a Feedback Customer – Submit Driver Rating Driver – View Driver Rating Announcement Page Final Report | 33.33% |

1. Constraints and Risks (Irdi)

* Were there any social, ethical, policy, or legal constraints?
* Did you have access to the data, services, and resources you needed?
* Was there anything else you needed?

1. Reflection (Irdi just see what you can do about this part. Melvir/Giulio will add on later)

* What were the lessons learned from doing this project?
* What went well?
* Which of your practices would you consider to be best practices?
* What didn't go well?
* What isn’t working, and how did you work around it?
* For the features that were not implemented, what were the issues?

Recommendations

* What would you do differently?
* What advice do you have for other teams?