Introduction

Remember that the learning outcomes for the project component are (taken from Project_descriptions_22-20.pdf on Canvas):

- integration (of knowledge and datasets) to solve the problem at hand,
- self-learning of new topics as needed,
- project and timeline management, including simplifying the problem or breaking it down to smaller manageable tasks,
- communication and collaboration within a team and between teams, and
- solving a problem for the user, not merely to our satisfaction.

Below are the dimensions on which we grade each component, and *some* examples of the evidence we look for in each of those dimensions.

Video Presentation (10%)

	Low score	**High score**
Script content/clarity	- Unclear/Ambiguous description of work done. - Assertions made are not correctly/well justified by evidence. - Too much "smoke"/"sales pitch" content e.g. promise of results to come. - Presentation does not flow well.	- Script is focused and efficient in delivering information about the project. There is a clear structure and flow to delivering the intended outcome and result. - Sincere and truthful representation of work that has been done. - Easy to understand flow of presentation and transition of sections - Final message/takeaway is clear and easy to grasp.
Visuals	- Slides are riddled with bullet points. - Poor choice of graphics/charts.	- Appropriate choice of charts/graphs/diagrams to convey information.
Appropriateness for audience	- Presentation is simply a summary of work done, with no awareness or empathy for the specific audience.	- Technical terms explained where necessary. - Technical content appropriate for audience. - Awareness of user/audience reflected in structure of sections and in how findings and the results are pitched.
Creativity	- Delivery is monotonous, inviting viewer to watch at 2x or 3x.	- Effort to use novel tools/structure of video. - Originality/innovation in capitalising on medium. - Engaging tone, passionate in delivery.

Journal (15%)

Information in the journal will be used to understand how the project was managed, what individual contributions were, and how you were able to assimilate the information in the course (and other courses) into the execution of the project.

	Low score	**High score**
Peer review		- Precise summary of individual roles, strengths and weaknesses with evidence. - If possible, tie back to data scientist roles discussed in lectures and by guest speakers.
Meeting minutes		- Information on attendees, dates, main discussion and action items for **three** meetings.
Content	- Absence of clear sections and a contents page. - Each component is written in a long paragraph, like a diary entry. - Throughout the report, bullet points are used instead of complete sentences. - Grammatical/spelling errors present.	- Report has a clear structure, broken up into sections and paragraphs within sections. - Written in good English (complete sentences).
Reflections	- Perfunctory connections to CS1010 and DSA1101. - No evidence of self-directed learning.	- Connections drawn to experiences and tasks outside of school, e.g. using knowledge in hackathons or other classes, or using git for other classes and projects, or experiences here in DSA3101 triggering a deep interest in particular topics. - Identification of how topics in the course shaped the outputs of the project, e.g. the presentation, codes written, models used, etc. - Outline of learning done on your own, and a reflection/recognition of your strengths/weaknesses.
Project timeline	- Completed tasks briefly described in words. - Little awareness of workflow as related to data science life cycle.	- A Gant chart depicting planned and eventual timelines and milestones - Description of delegation of duties and roles and project tracking plan.

		- Reflection on how the following aspects were handled: --- initial expectations of team-members, --- unexpected delays and issues --- communication issues --- working relationships within and between teams. - Demonstrate awareness of phases of project within data science life cycle.

Tech Report (15%)

The technical report is meant to be a document that you could pass to your successor. Upon reading your report and having been given access to your repository, this person should be able to continue your work, or at least reproduce your results.

	Low score	**High score**
Code explanations	- Use of screenshots for code	- Provides an overview of repo structure. - Contain instructions on running models, testing code. - Highlight possible issues, e.g. with versions/OS or bugs.
Structure/Language	- Demonstrates poor understanding of work completed by sub-team within the project. - Not a self-contained report; requires substantial additional readings to get started with using the product. - Every section is a list of bullet points. - Appears to be a listing of functions written, with no further explanations.	- Good use of visuals or diagrams, e.g. to explain overall architecture. - Written in proper English (complete sentences). - Overview section provides information on the document, allowing a user to find a particular section he/she is looking for easily.
Back-End Team	- Pure focus on model accuracy. - No indication of why models were shortlisted; gives an impression that models were swapped in/out with the hope of improving accuracy. - No references listed.	- Literature review of existing methods - Justification for models/solution chosen, probably by tying back to user interviews from Front-End team. - In-depth study of performance of model and it's failings. - Interpretation of model. - Demonstrate an awareness of how model can be tracked and improved after deployment.
Front-End Team	- No F2F interviews conducted, only mass surveys sent out. - Functionality of interface is very basic, laggy (not responsive) and/or contains bugs. - Little originality in interface.	- ~5 Interviews described briefly; results summarised, indicating how it led to refinements in solution. Repeat consults with user. - If no suitable user could be identified, describe how research was done on your user persona.

	- Charts used are not multivariate; simply use stock chart types, e.g. out-of-the-box bar charts and line charts.	- Wireframes to High-fidelity application evolution demonstrated. - Justification for choices link back to lecture on design, or from other readings. - Aesthetically pleasing Front-end created, using HTML/CSS customisation where appropriate. - Effort to create custom visualisation/charts/graphics, e.g. high-dimensional charts that convey insights better.

Project Success (30%)

While there is no specific submission related to this component, information from the video, the two reports, my interactions with your group, code on github and github commits (# of commits, exact lines committed etc.) will be used as input for these scores. Within this component, the frontend-backend sync will probably be common to both sub-teams, but other components are specific to the sub-teams.

	Low score	High score
Deployment	- "docker compose" does not work. - Services not dockerised e.g. model exists as a static file that the front-end loads.	- Entire solution is Dockerized, "docker compose" works. - Front-end communicates with back-end through Flask or similar architecture that provides separation of the two, and the ability to scale up either one. - Sub-team has experimented with different containers/services not covered in class e.g. databases for storing data. - Use of Docker demonstrates improved comfort with Linux/bash shell scripting.
Code quality	- Functions are not well-documented - All code exist in exploratory Jupyter notebooks or Rmd files, not as scripts. - Individuals create and merge their own pull requests, or no pull requests are used.	- A README file exists in most directories, to get a user started. - Classes defined where appropriate. - Unit testing is used in repo, to allow new contributions to be made easily without breaking old functionality. - Pull requests used to check and review each other's code.
Front-Back Sync	- Front and Back ends do not meet regularly, or user interviews are not fed back to the back-end team properly, or do not feed back into the final solution for the user. - Two teams appear to be working on separate assumptions or understanding of the problem.	- Evidence of constant communication between the two sub-teams. - Conflicts between two sub-teams are resolved amicably, e.g. --- regarding jurisdiction. --- deadlines and/or milestones revised to accommodate problems in sub-teams.

	- One sub-team continues to work as initially planned even though the other sub-team cannot deliver what was initially promised.	- Both teams help each other out to complete tasks before deadlines.
Project Progress	- Little forward planning in terms of the end goal before the semester ends, or how this product can be continued after the project. - Each individual in the team worked on separate tasks independently and then results put together at the end.	- Sub-teams demonstrate ability to redefine the problem independently and to break it into smaller problems before tackling them. - Resourcefulness to collect additional data on your own, or to collect, clean and tag a dataset for your own use. - Team works cohesively together, providing inputs and feedback to one another to create a better solution.