Verification and Validation Report:

April 3, 2024

1 Revision History

Date	Version	Notes
March 6, 2024	1.0	Upload VnVReport
April 2, 2024	2.0	Update VnVReport with feedback
April 3, 2024	2.1	Label tables and better organize information

2 Symbols, Abbreviations and Acronyms

symbol	description
ObjectID	MongoDB Record/Document Id
DB	Database
Schedule	Model type for MongoDB Schedule Collection
Command	Model type for MongoDB Command Collection
VM	Virtual Machine
NEUDOSE	NEUtron DOSimetry Exploration
UI/UX	User Interface/User Experience
JSON	JavaScript Object Notation

Contents

List of Tables

List of Figures

3 Functional Requirements Evaluation

3.1 MCT Application Accessibility

Table 1: MCT Application Accessibility Tests

Test Id	**Notes**	**Result**
FR-SLN1	MCT application is a site hosted on Netlify which was accessed manually using Safari, Chrome and Firefox	Pass
FR-SLN2	Manually established a connection on a TCP port and sent linux-commands through this port	Pass

3.2 Managing User Roles

Table 2: Managing User Roles Tests

Test Id	**Notes**	**Result**
FR-SLN3	Manually created a new user through the MCT's frontend facing application	Pass
FR-SLN4	Manually edited and removed a user through the MCT's frontend facing application	Pass

3.3 Scheduling and Executing Commands

Table 3: Scheduling and Executing Commands Tests

Test Id	**Notes**	**Result**

FR-SLN5	The addition, modification, and deletion of command sequences have been tested manually through the frontend.	Pass
FR-SLN6	Manually executed a scheduled command sequence and viewed output through the frontend.	Pass

3.4 Cancelling Commands

Table 4: Cancelling Commands Tests

Test Id	**Notes**	**Result**
FR-SLN7	The cancellation of command sequences have been tested manually through the frontend.	Pass

3.5 Validating Scheduled Commands

Table 5: Validating Scheduled Commands Tests

Test Id	**Notes**	**Result**
FR-SLN8	Manually tested through the frontend.	Pass

3.6 Permission List Criteria for User

Table 6: Permission List Criteria for User Tests

Test Id	**Notes**	**Result**

FR-SLN9	Manually tested execution of invalid command sequence through the frontend.	Pass

3.7 Permission List Criteria for Command Target

Table 7: Permission List Criteria for Command Target Tests

Test Id	**Notes**	**Result**
FR-SLN11	Manually tested execution of invalid command sequence through the frontend.	Pass

3.8 Managing Scheduled Command Sequences

Table 8: Managing Scheduled Command Sequences Tests

Test Id	**Notes**	**Result**
FR-SLN12	Manually tested managing (adding, deleting, editing) list of command sequences.	Pass

3.9 Selecting and Editing Satellites

Table 9: Selecting and Editing Satellites Tests

Test Id	**Notes**	**Result**

FR-SLN13	Manually tested managing (adding, removing) satellites of interest through the frontend.	Pass

3.10 Viewing Configured Satellites

Table 10: Viewing Configured Satellites Tests

Test Id	**Notes**	**Result**
FR-SLN14	Manually tested through the frontend.	Pass

New changes to FR-SLN14 have been made to test the fetching of the data from external libraries (instead of testing for correctness). These tests includes fetching the data based on the state of the TLE passed in, Valid TLE, Invalid TLE, Valid Start Date, Invalid State Date, and a combination of both.

3.11 Detecting Satellite and Scheduling Command

Table 11: Detecting Satellite and Scheduling Command Tests

Test Id	**Notes**	**Result**
FR-SLN16	Unit tests hav been written to only executed on the date and time range specified for an overpass date.	Pass

4 Nonfunctional Requirements Evaluation

4.1 Usability

Table 12: Nonfunctional Requirements Evaluation Tests

Test Id	**Result**
usability-1	Pass
usability-2	Pass

The tests above were conducted in two separate usability testing demonstrations, involving members from NEUDOSE's MIST team. All users involved in testing had no prior experience with the application, which focuses on their ability to learn and navigate the system independently. Both usability-1 and usability-2 had surveys to collect feedback for enhancement in the application's design and functionality.

The feedback that we received from usability-1 the second usability test, usability-2, which helped assess the application's usability.

Overall, users found the application intuitive to use while pointing out some areas for improvement. The data for this survey is attached below in Section ADD. In our first usability demonstration, we received a score of 3/5. Then after re-iterating our frontend design, we received a usability score of 5/5 from our stakeholders. This score measures how intuitive and easy it was for new users to identify core functionalities in the web application. Additionally, from usability-2 we found that it takes users on average less than five minutes to identify and explore the core use cases of the application.

usability-3 - NFR 10.5

- After careful consideration, our team has decided not to proceed with the implementation of the accessibility-focused verification and validation (VnV) plan, as outlined under the Usability-3 category, NFR: 10.5. This decision reflects our current prioritization of resources and development efforts.

- While we recognize the importance of accessibility in software development, our decision to forego this specific aspect of testing at this time allows us to allocate our resources towards other critical areas of development. We remain committed to creating an inclusive and accessible application and plan to revisit and incorporate comprehensive accessibility features and testing in future development phases.

4.2 Performance

Table 13: Performance Tests

Test Id	**Result**
performance-2	Pass
performance-4	Pass

For performance-2, we have deployed our application using external cloud providers. We use Netlify to host our frontend facing application and a DigitalOcean virtual machine to host our backend and server-side logic. In addition, we rely on the monitoring tools offered by these external providers to check the system's availablity and usage at regular intervals. The figures provided below illustrate the application's bandwidth, and CPU usage in the last hour. This time frame can also be specified for a longer duration. From the images below, our application is not very resource intensive where CPU usage is less than 1%. The continuous monitoring tools configured on both DigitalOcean and Netlify provides notifications on the application's uptime and downtime. Since the deployment of the application, the system's overall uptime has been 100% excluding scheduled maintenance periods. This is consistent with the availability that these cloud poviders offer, where both DigitalOcean and Netlify guarantees an uptime of 99.99% for deployments.

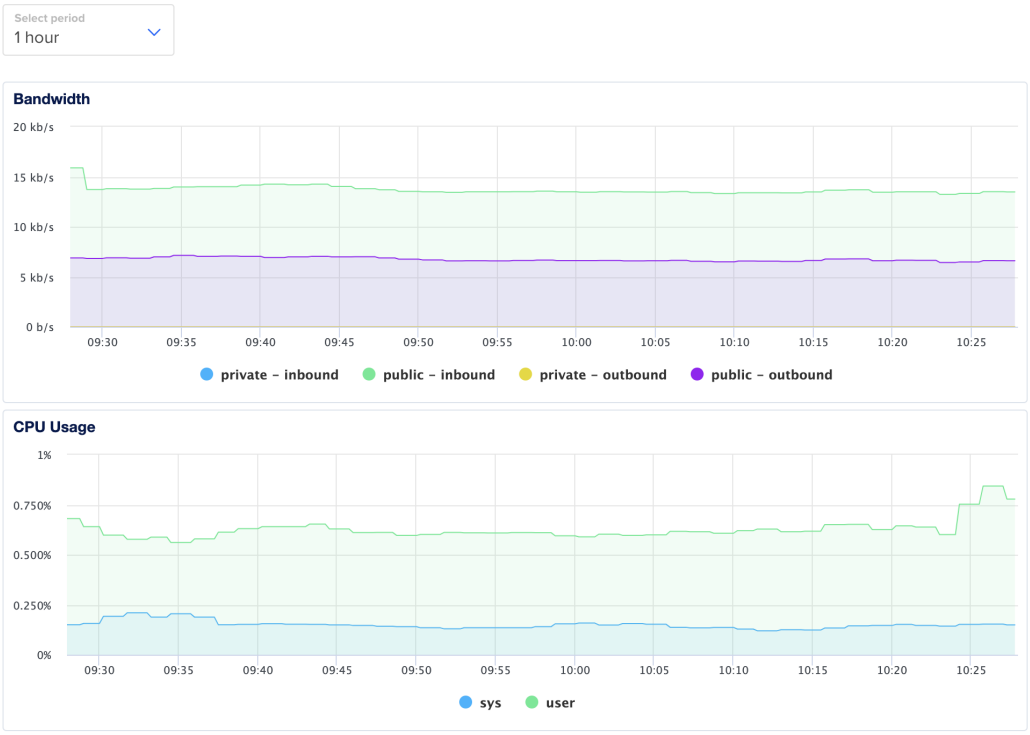


Figure 1: Application usage in last hour

For performance-4, scenarios which cause internal exceptions have been identified and implemented in automatic test cases. These automated tests have been implemented using Jest.js. Several of these tests involve handling edge cases which aim to catch internal exceptions. The figure below showcases the results from running all 88 unit tests pertaining to the functionality of the Satellite, Schedule, Command, and User modules.

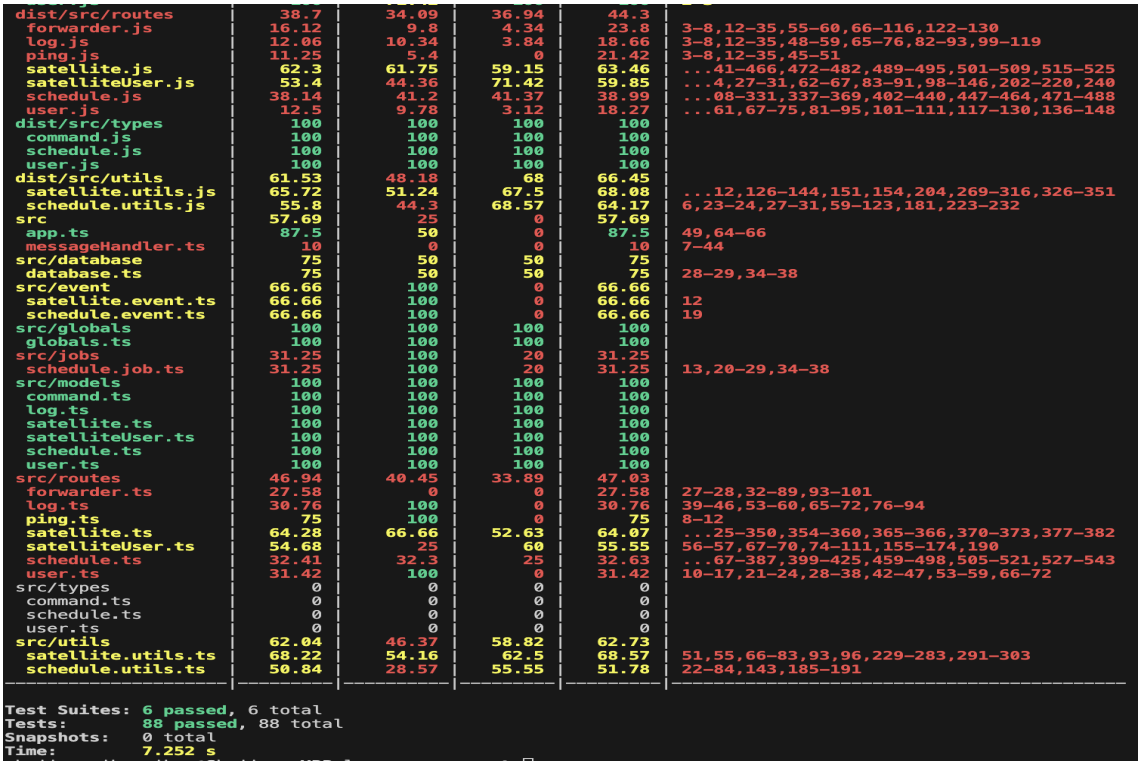


Figure 2: Jest.js Unit Test Results

4.3 Environmental

Table 14: Environmental Tests

Test Id	Result
environment-1	Pass
environment-2	Pass
environment-3	Pass

For environmental-1, the application is setup to be hosted on Netlify's free service at the current moment. This applies that the features are also limited and that there cannot be a lot of traffic sent which makes testing the

load much challenging. However, JMeter was used to verify the capabilities and challenged the application with 20 virtual users for 20 mintutes. Here are the results:

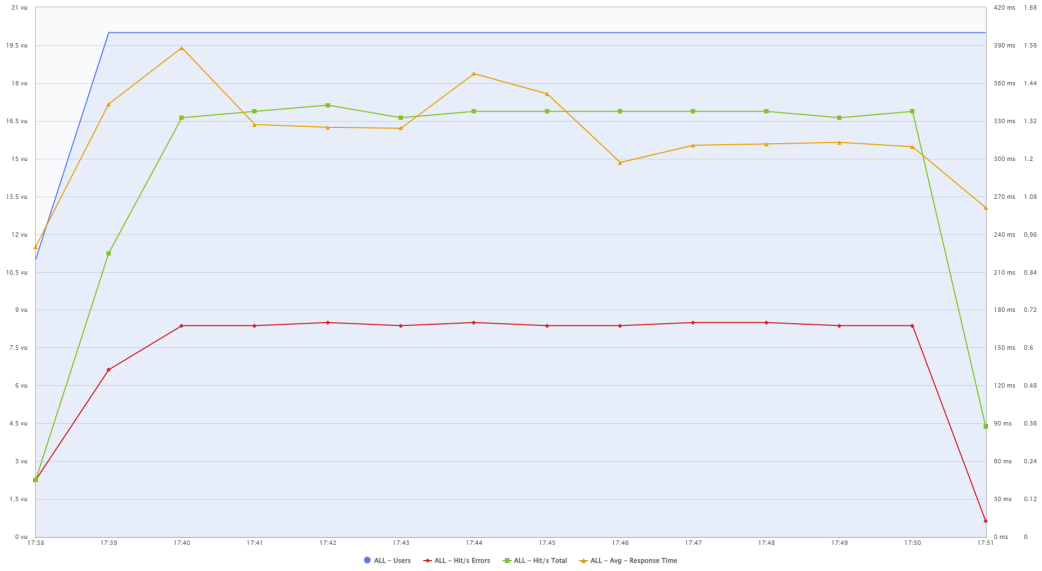


Figure 3: Stress test - JMeter 20 Mock Users

For environmental-2, we leveraged Netlify's powerful development platform. Netlify facilitated the creation of a self-contained and fully functional development environment, streamlining the workflow and significantly enhancing productivity. The developers were able to set up their local environments effortlessly, following the instructions provided, and could launch the development setup with confidence, knowing it would initialize and run without errors.

For environmental-3, we are using GitHub issues to keep track of the dates and time for the project. Issues are created for the tasks that are remaining to be performed along with attaching the development branch that focuses on it. This way, we are able to ensure that the functional, non-functional and outside requirements are being fulfilled on-time.

4.4 Maintenance

Table 15: Maintenance tests

Test Id	**Result**
maintenance-1	Pass
maintenance-2	Pass
maintenance-3	Pass
maintenance-4	Pass
maintenance-5	Pass

For maintenance-1, this system test was manually tested through the frontend. Upon executing a command, logs of a command are stored in MongoDB as JSON objects instead of plain or text files.

For maintenance-2, GitHub Actions is correctly configured for the project, triggering automated build and test checks. The test checks are triggered every commit, and every PR, seamlessly integrated in the Github repository.

For maintenance-3, the system identifies an outdated version automatically

by utilizing node's npm install command before the server is ran, and applies any changes (upgrades and downgrades). In addition, deliberate code formatting issues were introduced, and automatically corrected by esLint, the chosen code linter.

For maintenance-4, the project directory structure is intuitive, code is divided into client and server folders (front-end and back-end), with components, styles, and pages being divided for logical ordering. Variable, function, and template names follow consistent naming schemes, and are grouped where needed. Lastly, rigorous code reviews are performed to maintain the quality of the repository.

For maintenance-5, A code formatting tool, ESLint is integrated in the development environment through Visual Studio Code, and is enforced in code review. Since it is automatically integrated, random code files which are inspected match indentation, variable naming, and code structure.

4.5 Integrity

Table 16: Integrity Tests

Test Id	**Result**
integrity-5	Pass
integrity-6	Pass
integrity-7	Pass

For integrity-5, various simulated satellite failures have been implemented in test cases to test and monitor error messages which are consistent amongst other messages, and communicate the nature of missing data attributes.

For integrity-6, the application performs essential functionalities on all browsers (Safari, Chrome, Firefox).

For integrity-7, the typescript compiler verifies HTML and CSS validity, and

has not identified any compatibility issues across devices. In addition, esLint is applied to apply HTML and CSS coding standards when it comes to formatting. The application user interface remains consistent and functional across different devices and browsers.

4.6 Privacy and Accessibility

Table 17: Privacy and Accessibility Tests

Test Id	**Result**
access-1	Pass
access-2	Pass
privacy-1	Pass

For access-1, the application integrates with an external authentication service (Auth0) to manage user authentication and account management. New and existing users will be registered and logged in using this service The test cases for this requirement are AM2 and AM3 and it can be found below in section 8 of Regression Testing.

For access-2, the application integrates with an external authentication service (Auth0) to manage user authentication and account management. This service allows an administrator to configure a rate limit of 10 login attempts per minute. As a result, the application enforces a 1 minute timeout period before the user can attempt to authenticate again. The test case for requirement is AM1 and it can be found below in section 8 of Regression Testing.

For privacy-1, the test case for this requirement has not been implemented yet. Refer to test case REG-SAT1 in section 8 of Regression Testing.

5 Unit Testing

5.1 Scheduling Module

5.1.1 API Endpoints

POST /schedule/createScheduledCommand

Table 18: Unit Tests for
POST /schedule/createScheduledCommand

Id	Reference Req. Id	Input	Expected Output	Actual Output	Result
SM1	FR-12, FR-13, FR-17, FR-16	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	Pass
SM2	FR-17, FR-16, FR-13	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “commandNotInCriteria”	status: 500, Error: ”Invalid command sequence or user permissions”	status: 500, Error: ”Invalid command sequence or user permissions”	Pass

PATCH /schedule/updateScheduledCommand

Table 19: Unit Tests for
PATCH /schedule/updateScheduledCommand

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM3	FR-3, FR-4, FR-5	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “start”	Pass
SM4	FR-11	userId: ObjectID, satelliteId: ObjectID, commandId: ObjectID, command: “commandNotInCriteria”	status: 500, Error: ”Invalid command sequence or user permissions”	status: 500, Error: ”Invalid command sequence or user permissions”	Pass

GET /schedule/getSchedulesBySatellite

Table 20: Unit Tests for
GET /schedule/getSchedulesBySatellite

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**

SM5	FR-13	satelliteId: ObjectID	Message: "Fetched schedules", schedules: Schedule[]	Message: "Fetched schedules", schedules: Schedule[]	Pass
SM6	FR-13	satelliteId: ObjectID, status: "PASSED"	Message: "Fetched schedules", schedules: Schedule[]	Message: "Fetched schedules", schedules: Schedule[]	Pass

GET /schedule/getCommandsBySchedule

Table 21: Unit Tests for
GET /schedule/getCommandsBySchedule

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM7	FR-13	satelliteId: ObjectID	Message: "Fetched commands", commands: Command[]	Message: "Fetched commands", commands: Command[]	Pass

DELETE /schedule/deleteScheduledCommand

Table 22: Unit Tests for
DELETE /schedule/deleteScheduledCommand

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM8	FR-13	commandId: ObjectID, userId: ObjectID	Message: "Success. Deleted schedule"	Message: "Success. Deleted schedule"	Pass
SM9	FR-11	commandId: ObjectID, userId: ObjectID	status: 500, Error: "Insufficient permissions"	status: 500, Error: "Insufficient permissions"	Pass

GET /satellite/getSatelliteInfo

Table 23: Unit Tests forGET /satellite/getSatelliteInfo

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM10	FR-15		status: 200	status: 200	Pass

GET /getPolarPlotData

Table 24: Unit Tests forGET /getPolarPlotData

Id	Reference Req. Id	Input	Expected Output	Actual Output	Result
SM11	FR-15	startTime = 2024-01-06T10:15:00Z; endTime = 2024-01-06T10:22:00Z;	status: 200	status: 200	Pass
SM12	FR-15	startTime = 2024-01-06T10:15:00Z; endTime = ””;	status: 500	status: 500	Pass

GET /maxElevation

Table 25: Unit Tests forGET /maxElevation

Id	Reference Req. Id	Input	Expected Output	Actual Output	Result
SM13	FR-16	startTime = 2024-01-06T10:15:00Z; endTime = 2024-01-06T10:22:00Z;	status: 200	status: 200	Pass

SM14	FR-16	startTime = 2024-01- 06T 10:15:00Z; endTime = ””;	status: 500	status: 500	Pass

GET /getNextPasses

Table 26: Unit Tests forGET /getNextPasses

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM15	FR-15	TLE = 59909	status: 200	status: 200	Pass
SM16	FR-15	TLE = ””	status: 500	status: 500	Pass

GET /getSolarIllumination

Table 27: Unit Tests forGET /getSolarIllumination

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM16	FR-15	TLE = 59909	status: 200	status: 200	Pass
SM17	FR-15	TLE = ””	status: 500	status: 500	Pass

POST /changeTLE

Table 28: Unit Tests forPOST /changeTLE

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM18	FR-14	TLE = 59909	status: 200	status: 200	Pass
SM19	FR-14	TLE = ""	status: 500	status: 500	Pass

5.2 Satellite Users Module

5.2.1 API Endpoints

POST /satelliteUser/createSatelliteUser

Table 29: Unit Tests for
POST /satelliteUser/createSatelliteUser

Id	**Input**	**Expected Output**	**Actual Output**	**Result**
SUM1	userId: ObjectID, satelliteId: ObjectID, adminId: ObjectID, validCommands: ["teardown"]	satelliteUserId: ObjectID, satelliteId: ObjectID, validCommands: ["teardown"], adminId: ObjectId	satelliteUserId: ObjectID, satelliteId: ObjectID, adminId: ObjectID, validCommands: ["teardown"]	Pass

SUM2	userId: ObjectID, satelliteId: ObjectID, adminId: "invalidAdminId", validCommands: ["teardown"]	status: 500, Error: "Invalid command sequence or user permissions"	status: 500, Error: "Invalid command sequence or user permissions"	Pass
SUM3	userId: ObjectID, satelliteId: "invalidSatelliteId", adminId: ObjectId, validCommands: ["teardown"]	status: 500, Error: "Invalid Ids"	status: 500, Error: "Invalid Ids"	Pass
SUM4	userId: ObjectID, satelliteId: ObjectID, adminId: ObjectId, validCommands: ["invalidCommand"]	status: 500, Error: "Invalid command sequence or user permissions"	status: 500, Error: "Invalid command sequence or user permissions"	Pass
SUM5	userId: "invalidUserId", satelliteId: ObjectID, adminId: ObjectId, validCommands: ["teardown"]	status: 500, Error: "Invalid command sequence or user permissions"	status: 500, Error: "Invalid command sequence or user permissions"	Pass

PATCH /satelliteUser/updateByUser

Table 30: Unit Tests for
PATCH /satelliteUser/updateByUser

Id	**Input**	**Expected Output**	**Actual Output**	**Result**
SUM6	satelliteUserId: ObjectID, satelliteId: ObjectID, adminId: ObjectID, validCommands: [“teardown”]	satelliteUserId: ObjectID, satelliteId: ObjectID, validCommands: [”teardown”], adminId: ObjectId	satelliteUserId: ObjectID, satelliteId: ObjectID, adminId: ObjectID, validCommands: [“teardown”]	Pass
SUM7	satelliteUserId: ObjectID, satelliteId: ObjectID, adminId: ”invalidAdminId”, validCommands: [“teardown”]	status: 500, Error: ”Invalid command sequence or user permissions”	status: 500, Error: ”Invalid command sequence or user permissions”	Pass
SUM8	satelliteUserId: ObjectID, satelliteId: ”invalidSatelliteId”, adminId: ObjectId, validCommands: [“teardown”]	status: 500, Error: ”Invalid Ids”	status: 500, Error: ”Invalid Ids”	Pass

SUM9	satelliteUserId: ObjectID, satelliteId: ObjectID, adminId: ObjectId, validCommands: [“invalidCommand”]	status: 500, Error: ”Invalid command sequence or user permissions”	status: 500, Error: ”Invalid command sequence or user permissions”	Pass
SUM10	satelliteUserId: ”invalidSatelliteUserId”, satelliteId: ObjectID, adminId: ObjectId, validCommands: [“teardown”]	status: 500, Error: ”Invalid command sequence or user permissions”	status: 500, Error: ”Invalid command sequence or user permissions”	Pass

GET /satelliteUser/getUserBySatellite

Table 31: Unit Tests for
GET /satelliteUser/getUserBySatellite

Id	**Input**	**Expected Output**	**Actual Output**	**Result**
SUM11	satelliteId: ObjectID	Message: “Fetched satellite users”, satelliteUsers: satelliteUsers[]	Message: “Fetched satellite users”, satelliteUsers: satelliteUsers[]	Pass

SUM12	satelliteId: "invalidSatelliteId"	Message: "Fetched satellite Users", satelliteUsers: []	Message: "Fetched satellite Users", satelliteUsers: []	Pass

GET /satelliteUser/getCommandsBySatelliteAndUser

Table 32: Unit Tests for
GET /satelliteUser/getCommandsBySatelliteAndUser

Id	**Input**	**Expected Output**	**Actual Output**	**Result**
SUM13	satelliteId: ObjectID, userId: ObjectId	Message: "Fetched satellite users", satelliteUsers: satelliteUsers[]	Message: "Fetched satellite users", satelliteUsers: satelliteUsers[]	Pass
SUM14	satelliteId: "invalidSatelliteId", userId: ObjectId	Message: "Fetched satellite Users", satelliteUsers: []	Message: "Fetched satellite Users", satelliteUsers: []	Pass

SUM15	satelliteId: ObjectId, userId: "InvalidUserId"	Message: "Fetched satellite Users", satelliteUsers: []	Message: "Fetched satellite Users", satelliteUsers: []	Pass

DELETE /satelliteUser/deleteByUser

Table 33: Unit Tests for DELETE /satelliteUser/deleteByUser

Id	**Input**	**Expected Output**	**Actual Output**	**Result**
SUM16	adminId: ObjectID, satelliteUserId: ObjectId	Message: "Removed User from satellite"	Message: "Removed User from satellite"	Pass
SUM17	adminId: "invalidAdminId", satelliteUserId: ObjectId	Message: "Invalid Ids"	Message: "Invalid Ids"	Pass
SUM18	satelliteUserId: "invalidId", adminId: ObjectId	Message: "Invalid Ids"	Message: "Invalid Ids"	Pass

5.3 Helper Functions

executeScheduledCommands(satelliteId: ObjectID, scheduleId: ObjectID) ⇒ void

Table 34: Unit Tests for executeScheduledCommands()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM20	FR-10	satelliteId: ObjectID, scheduleId: ObjectID	List of log records corresponding to the executed command records for the specified scheduleId	List of log records corresponding to the executed command records for the specified scheduleId	Pass

rescheduleLeftoverCommand(satelliteId: ObjectID, scheduleId: ObjectID) ⇒ void

Table 35: Unit Tests for rescheduleLeftoverCommand()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM21	FR-10, FR-11	satelliteId: ObjectID, scheduleId: ObjectID	Schedule specified in the request has no commands with status: status: "QUEUED"	Schedule specified in the request has no commands with status: status: "QUEUED"	Pass

addSchedulesForNext7Days(satelliteId: ObjectID, noradId: number) ⇒ void

Table 36: Unit Tests for addSchedulesForNext7Days()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM22	FR-16, FR-17	satelliteId: ObjectID, noradId: number	Satellite specified in the request has new schedules for the next seven days	Satellite specified in the request has new schedules for the next seven days	Pass

getSatelliteInfo(date: Date, tleLine1: number, tleLine2: number) ⇒ { positionEci: number, velocityEci: number, longitude: number, latitude: number, height: number, azimuth: number, elevation: number, rangeSat: number }

Table 37: Unit Tests forgetSatelliteInfo()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**

SM23	FR-15	new Date(), 1 55098U 23001CT 23359.66872105 .00021921 00000-0 89042-3 0 9991, 2 55098 97.4576 58.0973 0014812 57.5063 302.7604 15.24489013 54199	{ positionEci: number, velocityEci: number, longitude: number, latitude: number, height: number, azimuth: number, elevation: number, rangeSat: number }	{ positionEci: number, velocityEci: number, longitude: number, latitude: number, height: number, azimuth: number, elevation: number, rangeSat: number }	Pass
SM24	FR-15	"", 1 55098U 23001CT 23359.66872105 .00021921 00000-0 89042-3 0 9991, 2 55098 97.4576 58.0973 0014812 57.5063 302.7604 15.24489013 54199	Error: Invalid Date	Error: Invalid Date	Pass

SM25	FR-15	new Date(), "", ""	Error: Invalid TLE	Invalid TLE	Pass

isSunLit(date: Date, lon: number, lat: number, height: number) ⇒ boolean

Table 38: Unit Tests for isSunLit()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM26	FR-15	new Date(), 0, 0, 0	isDefined	isDefined	Pass
SM27	FR-15	"", 0, 0, 0	isNotDefined	isNotDefined	Pass
SM28	FR-15	new Date(), 0, 0, 20000	Error: Height must be in km	Error: Height must be in km	Pass

setTLE(tle: string) ⇒ void

Table 39: Unit Tests for setTLE()

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
SM29	FR-14	"55098"	resolves	resolves	Pass
SM30	FR-14	"abcd"	Error: Invalid TLE	Error: Invalid TLE	Pass

SM30	FR-14	""	Error: Empty TLE	Error: Empty TLE	Pass

6 Regression Testing

6.1 Authentication Module

Table 40: Regression Tests for Authentication Module

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
AM1	NFR 14.1.2	email: "test1@ gmail.com" password: "correct" Repeat 10 times	Disable login functionality for one minute	Disable login functionality for one minute	Pass
AM2	NFR 14.1.1	email: "test1@ gmail.com" password: "correct"	Successfully log user into the application's home page	Successfully log user into the application's home page	Pass

AM3	NFR 14.1.1, NFR 14.1.2	email: “test1@ gmail.com” password: “correct”	Successfully register user into the application and directs them to the home page	Successfully register user into the application and directs them to the home page	Pass

6.2 Satellite Module

Table 41: Regression Tests for Satellite Module

Id	**Reference Req. Id**	**Input**	**Expected Output**	**Actual Output**	**Result**
REG-SAT1	NFR 14.3.1	port: number	Socket successfully connected with hash output	Socket successfully connected with hash output	TBD

7 Changes Due to Testing

7.1 Feedback from Revision 0

During the Revision 0 demonstration, the application was assessed on its functionality and overall usability. The major feedback received was the lack of integration between the different functional components. For example, both user interfaces and endpoints were created to add and update valid commands, however these two components were not integrated together. This has

been resolved by integrating the frontend and backend components relating to managing satellite information and valid commands for an operator. As a result, users are able to interact with the application without intervention from a developer.

7.2 Feedback from Usability Testing

Two usability tests have been conducted to assess the application. Each test involved stakeholders as well as new users from McMaster's NEUDOSE team to evaluate the core functionalities of the web application.

From the first usability testing demonstration, the critiques mainly highlighted issues with the UI/UX of application rather than the functionality. Specifically, users mentioned that the presentation of information for upcoming satellite overpasses felt cluttered and suggested to separate information related to satellite into different sections. After reviewing the feedback from this usability test, changes to the user interface was made such that overpasses for a satellite was presented in a tabular format as opposed to a card. This improved the readability and organization of data in the frontend.

Furthermore, heading into the second usability testing demonstration with member of the NEUDOSE team, all usability issues described above had been addressed. During this demonstration, users responded positively to the new UI/UX changes, however, there were some issues when loading changes to satellite information in the frontend. This has been resolved by refactoring how the information from the backend is captured in the frontend. To address this, an external query management package known as TanStack Query was integrated into the frontend facing application to manage requests sent to the backend. This provided the ability to automatically react and load data from the backend, ensuring data changes are timely displayed. Additionally, users felt that as the number of schedules for a satellite increases, it would be difficult to find a particular schedule. This was addressed by adding a date filter for satellite passes, which enables users to search for schedules by specifying the start and end dates.

7.3 Additional Application Changes

Although FR-SLN2 was verified locally, the initial deployment of the application's backend did not support evaluating this test in production. When testing FR-SLN2 on the deployed backend of the application, there were issues establishing a TCP connection. For some context, the backend application was deployed using DigitalOcean's App Platform which is a containerized service. After investigating, this deployment had very limited customizing features. In particular, DigitalOcean's App Platform did not offer the ability to open a port to establish a TCP connection. As a result, a connection could not be made with the deployment build and FR-SLN2 was verified not able to be tested in production. In order to resolve this, a virtual machine (VM) had been provisioned on DigitalOcean. This VM was then configured to run the backend application. Since a VM has customizable options, a TCP port was configured which was then used to establish a valid TCP connection. Finally, the FR-SLN2 functional requirement test was verified with this new deployment.

7.4 Changes to the VnV Plan

FR-SLN15

- This test has been removed as the stakeholders no longer requested this requirement.

performance-3 (NFR 11.3)

- Precision calculation will be tested by external library providers as the calculations are no longer in the scope of the application.

access-1 (NFR 14.1.1)

- NFR 14.1.1 has been revised and the application will only require to support Single-Sign-On (SSO) authentication instead of Multi-Factor-Authentication (MFA).

8 Automated Testing

For the frontend repository, leo-client-app, we use Next.js and Netlify's built-in testing mechanism to ensure both local and production environments are

error-free. Locally, developers can run the command **npm run build** to create a build directory with a production build of the application. Developers can run this command to verify that their code has no production issues before pushing from their remote machine. The execution of this command is also automated via Github Actions anytime a commit is pushed into the repository as well as during merging from feature branches to the **main** branch of the repository. Since the main branch reflects the production environment, this build check must pass before a feature branch can be merged into the main branch. In addition, a Netlify build is created through Github Actions for every feature branch being merged into the main branch. Since Netlify is hosting the frontend-facing components of the application, this build must be successful before merging into the main branch.

For the backend repository, leo-server-app, tests are automated using Jest.js. All unit tests pertaining to the Satellite, Schedule, and Command modules are found in the src/tests folder. Through Github Actions, all tests are automatically executed when issuing a pull request to merge to the main branch. Developers can also automate this test by running the command, **npm run test** locally. This is also the same command Github Actions uses upon pushing a commit to a branch.

9 Trace to Requirements

Req. ID	**System Test ID**
FR-1	FR-SLN1, SM3
FR-2	FR-SLN2, FR-SLN3, FR-SLN4
FR-3	FR-SLN5, FR-SLN6
FR-4	FR-SLN5, FR-SLN6, SM3
FR-5	FR-SLN5, FR-SLN6, SM3
FR-6	FR-SLN7
FR-7	FR-SLN3, FR-SLN4, FR-SLN8
FR-8	FR-SLN8
FR-10	FR-SLN9, SM10

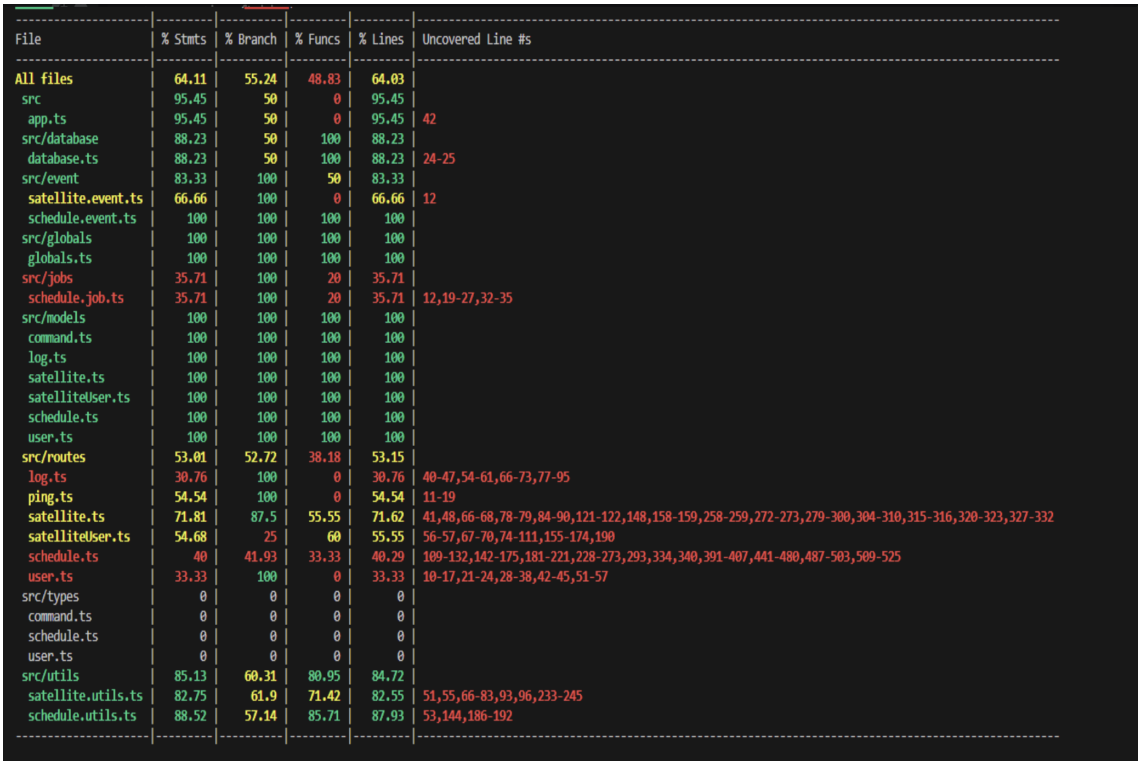
FR-11	FR-SLN9, SM4, SM9, SM11
FR-12	FR-SLN11, SM1
FR-13	FR-SLN12, SM5, SM6, SM7, SM8
FR-14	FR-SLN13
FR-15	FR-SLN14
FR-16	SM1, SM2, SM12
FR-17	FR-SLN16, SM1, SM2, SM12
NFR-10.1	usability-1
NFR-10.3	usability-2
NFR-10.5	usability-3
NFR-11.2	performance-2
NFR-11.4	performance-4
NFR-12.2.1	environmental-1
NFR-12.2.2	environmental-2
NFR-12.5	environmental-3
NFR-13.1.1	maintenance-1
NFR-13.1.2	maintenance-2
NFR-13.1.3	maintenance-3
NFR-13.1.4	maintenance-4
NFR-13.1.5	maintenance-5
NFR-13.2.1	support-1
NFR-13.2.2	support-2
NFR-13.2.3	support-3
NFR-14.1.1	access-1, AM2, AM3
NFR-14.1.2	access-2, AM1, AM3
NFR-14.2.5	integrity-5
NFR-14.2.6	integrity-6
NFR-14.2.7	integrity-7
NFR-14.3.1	privacy-1, REG- SAT1

10 Trace to Modules

Module	**System Test ID**
Authentication Module	AM1, AM2, AM3
Schedule Module	SM1, SM2, SM3, SM4, SM5, SM6, SM7, SM8, SM9, SM10, SM11, SM12
Satellite User Module	SUM1, SUM2, SUM3, SUM4, SUM5, SUM6, SUM7, SUM8, SUM9, SUM10, SUM11, SUM12, SUM13, SUM14, SUM15, SUM16, SUM17, SUM18
Satellite Module	REG-SAT1

11 Code Coverage Metrics

The below image details the code coverage for the leo-server-app repository. This provides metrics on the percentage of statements, branches, functions, and lines covered for each file.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	64.11	55.24	48.83	64.03	
src	95.45	50	0	95.45	
app.ts	95.45	50	0	95.45	42
src/database	88.23	50	100	88.23	
database.ts	88.23	50	100	88.23	24-25
src/event	83.33	100	50	83.33	
satellite.event.ts	66.66	100	0	66.66	12
schedule.event.ts	100	100	100	100	
src/globals	100	100	100	100	
globals.ts	100	100	100	100	
src/jobs	35.71	100	20	35.71	
schedule.job.ts	35.71	100	20	35.71	12,19-27,32-35
src/models	100	100	100	100	
command.ts	100	100	100	100	
log.ts	100	100	100	100	
satellite.ts	100	100	100	100	
satelliteUser.ts	100	100	100	100	
schedule.ts	100	100	100	100	
user.ts	100	100	100	100	
src/routes	53.01	52.72	38.18	53.15	
log.ts	30.76	100	0	30.76	40-47,54-61,66-73,77-95
ping.ts	54.54	100	0	54.54	11-19
satellite.ts	71.81	87.5	55.55	71.62	41,48,66-68,78-79,84-90,121-122,148,158-159,258-259,272-273,279-300,304-310,315-316,320-323,327-332
satelliteUser.ts	54.68	25	60	55.55	56-57,67-70,74-111,155-174,190
schedule.ts	40	41.93	33.33	40.29	109-132,142-175,181-221,228-273,293,334,340,391-407,441-480,487-503,509-525
user.ts	33.33	100	0	33.33	10-17,21-24,28-38,42-45,51-57
src/types	0	0	0	0	
command.ts	0	0	0	0	
schedule.ts	0	0	0	0	
user.ts	0	0	0	0	
src/utils	85.13	60.31	80.95	84.72	
satellite.utils.ts	82.75	61.9	71.42	82.55	51,55,66-83,93,96,233-245
schedule.utils.ts	88.52	57.14	85.71	87.93	53,144,186-192

Figure 4: Code Coverage Metrics

12 Appendix — Survey

12.1 Usability Survey Questions and Answers

Was the application intuitive for you to use? Rate from 1-5 from least to most intuitive.

- Usability Test 1: 3/5

- Usability Test 2: 5/5

Was any aspect of the application difficult to use? Provide any examples if you can.

- it's unclear how the scheduler works (i.e. you select a satellite then select a pass, then you can set commands but it's not obvious those commands have been succesfully assigned to that pass). It would be nice to be able to, from within the scheduling editor page, change which pass the schedule is being set up for.
- it would also be nice if once a set of commands is input for a pass the application gave some confirmation the schedule has been set and when it is set for.
- Not really, I thought everything was intuitive and easily navigable.

Did you encounter any bugs/errors/issues while you were using the application? Provide any examples if you can.

- i was only there for 10 minutes but nothing seemed to break. Not sure if setting the edit schedule with a new command (such as teardown) does anything yet, but if it's supposed to adjust the schedule the pass schedule does not seem to update with the new command
- The scheme could use some fixing (lightmode).

How clear and understandable was the content within the application? Were there any parts that confused you?

- Interface mostly makes sense! see previous comment on scheduling being unclear when the new schedule is set
- I think everything was clear and easily navigable.

Please provide your thoughts on the overall visual design of our application.

- Generally good, the white text on light background is somewhat hard to read (sattelites, schedule, logs, etc.)

- I think it looked nice, simple and easy to navigate. However, the area where it displayed the satellites command schedule felt a little too cluttered.

Are there any aspects of the UI that you found unappealing? Please explain why.

- The area where there it listed the next command schedule for satellites could benefit from a scroll bar.

How satisfied are you with the overall application? From a scale of 1-5 from least satisfied to most.

- First test: 3/5
- Second test: 4/5

What do you like most about the application? What do you like least?

- I liked how you show the schedule commands for the given week, etc. I also liked the extra information displayed for every satellite.

Are there any features/aspects of the application that you found useful or unnecessary? Please provide some examples.

- I think everything was very nice and nothing was redundant.

What part of the application in your opinion needs the most improvement. Why?

- The area where it displayed the scheduled commands is the biggest issue.

How can we improve the application to better meet your expectations?

- More organized (possibly add tabs?) and add scroll bar.

Is anything missing in your opinion? What would you add?

- Scroll bar for the commands and possibly make it less cluttered, a back button would be nice. Overall good job.

12.2 Usability Survey Notes and Summary

- The initial testing involved users with no prior experience with the application, focusing on their ability to learn and navigate the system independently. The feedback highlights both strengths and areas for enhancement in the application's design and functionality.
- Strengths Noted:
  - The satellite and overpass functionality received positive remarks for its quick response when navigating to specific longitudes and latitudes, indicating efficient performance in these areas.
  - The command deletion process within the application was described as "pretty solid," suggesting it is user-friendly and reliable.
- Areas for Improvement:
  - Visibility and Clarity: There was confusion about the "valid commands box," suggesting a need for increased visibility and clearer indications of its purpose. It's essential to make it more obvious to users what the valid commands are.
  - Logs Page Functionality: Questions were raised about the purpose of the logs page, with suggestions to include filters for better usability, indicating a need for clearer presentation and functionality.
  - User Interface (UI) Considerations:
    * The all-white interface led to suggestions for a light mode/dark mode feature, catering to different user preferences.
    * The addition of a feature to easily add satellites to the database directly from the first page was recommended.
    * Faster loading times for satellite changes were requested to enhance user experience.
    * Improvements in readability and presentation of information for upcoming satellite passes, possibly through a tabular format, were suggested to make the data more accessible.
    * A reevaluation of navigation within the schedule queue, seeking alternatives to horizontal scrolling, was advised.

* The introduction of a filter for satellite passes, enabling searches by date or specific passes, was recommended to improve usability.
* A call for a design that avoids the need for page scrolling, aiming to present all relevant information simultaneously.
* The presentation of information in blocks or separated formats was suggested to improve visual hierarchy and readability, particularly for satellite information.

References

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection. Please answer the following question:

1. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

- Quinn - The biggest difference in the way that the VnV plan was different from what was conducted was for me, the satellite calculations. the initial plan was to develop the code to calculate, however upon future research, was determined to be more accurate to rely on scientists and utilize open-source libraries instead. As a result, the VnV plan was modified to instead validate the flow of data, specifically the data and types of the parameters. This could have been anticipated earlier with given research, and more effort will be given in the future by researching before establishing a plan for testing.

- Umang - In terms of verifying and validating the functionality of the scheduling and authentication modules, we initially had wanted to automate both backend and frontend facing features using a unit-testing library. However, we instead decided to modularize the evaluation of the backend and frontend components. As a result, we opted for evaluating the backend API endpoints and helper functions using unit tests to measure the application's correctness. Then, we used acceptance testing with our stakeholders to assess the user interface of the application. Furthermore, this could have been anticipated by researching into common methods of testing server and client-side applications beforehand.

- Diamond - Reflecting on our initial Verification and Validation (VnV) Plan, we realized the necessity to deviate from certain aspects, notably

the Usability-3, NFR: 10.5, focused on assessing accessibility. This part of the plan, aimed at manually testing the application's accessibility for users with diverse needs, was ambitious in ensuring our system was universally accessible. However, due to constraints in resources, time, and the availability of a varied group of test participants with specific accessibility needs, we have decided not to proceed with this detailed testing approach. Instead, our efforts have pivoted towards leveraging best practices in accessibility design and relying on automated tools to assess some accessibility features. This decision underlines the often complex balance between ideal testing scenarios and the practical limitations of project environments. It also highlights the importance of flexibility in project management and the need to adapt plans based on available resources and evolving project dynamics. This experience reinforces the value of incorporating adaptability into our VnV strategies, allowing us to better anticipate and accommodate changes in future projects while still aiming to deliver high-quality, accessible software within our constraints.

- Rishi - When we look back at our initial plan for Verification and Validation (VnV) and compare it to what we actually ended up doing, we noticed quite a few differences. These changes were necessary because we encountered unexpected requirements and had to make practical adjustments to fit the scope and methods of our project. For example, we had to tweak our strategies to fit new tools and technologies, adjust to changes in our team's skills, and cope with shifts in project deadlines.

  These adjustments were all part of the unpredictable nature of software development. Plans that look good on paper often have to be adapted to deal with the real-world challenges we face, like technical hurdles, limited resources, and changes in project goals. Going through this process taught us the importance of being flexible and able to change direction when needed. While it's hard to predict every twist and turn, this project showed us that being adaptable is key to success.

  We learned that having a solid plan is essential, but so is being ready to adjust that plan as we go along and learn new things. This mindset will help us anticipate and handle changes better in future projects, making sure our VnV activities stay on track with what we're trying

to achieve and the limitations we face.

- Dhruv - The biggest difference in our initial plan and what ended up happening was the new changes and requirements that were added along the way. As a result, practical changes are a consequence especially if we want to fit it within our timeline and scope of our project. This is only natural in software development and was an important lesson to learn because no one can ever predict all incoming changes that will happen. This forced me to not get fixated on one solution and to keep thinking from different perspectives. Additionally, we were able to do user testing with our stakeholders which provided important and useful feedback. This motivated us to research more software testing methods, even the unconventional ones perhaps. As someone who was primarily working on the front-end, I never estimated that it would take as long as it did. It was an important lesson that every aspect of software development is vital.