U-Ride Ride-Hailing Management System

Sprint 1 Deliverables

Omar Abdelhady (3-bhd) Janna Osama (Jannaosama)
George Christo (georgechristo-ctrl) Farida Elanany (faridaelanany)
Abdelrahman Eid (Abdelrahman-233)

November 15, 2025

Repository: github.com/3-bhd/ride-hailing-app-team6

Contents

1 Sprint Overview

1.1 Sprint Goal

The goal of Sprint 1 was to build the foundation of the U/Ride Ride-Hailing Management System by implementing the core authentication and verification flows for the three user roles: passengers, drivers, and administrators. The main objective was to deliver a functional Minimum Viable Product (MVP) where users can register, log in, and interact with the system in a realistic and consistent manner. This sprint focused on establishing the essential backend logic, database setup, and interface structure required for the system to evolve in future sprints.

1.2 Scope of Sprint 1

Sprint 1 covered three user stories selected from the refined product backlog:

- **User Story 1 – Passenger Registration and Login:** Passengers are able to create an account, log in securely, and access a personalized dashboard to request rides.
- **User Story 4 – Driver Registration and Verification:** Drivers can complete a registration form, upload the required documents, and await approval from an administrator.
- **User Story 5 – Driver Online/Offline Status:** Approved drivers can log in to their dashboard and toggle their active status between online and offline.

These user stories establish the core onboarding and verification workflows and form the foundation upon which future ride-matching and trip-management features will be built.

1.3 Summary of Outcomes

By the end of Sprint 1, the team successfully delivered a working prototype that includes:

- Complete registration and login flows for passengers, drivers, and administrators.
- A functional **Passenger Dashboard** enabling authenticated passengers to request a ride.
- A full **Driver Registration** flow with secure document uploads.
- An **Admin Dashboard** that displays pending, approved, and rejected drivers and allows approval or rejection of applications.
- A **Driver Dashboard** that shows verification status, profile information, and an online/offline toggle.
- Integrated access-control logic ensuring that only authorized users may access specific pages.
- A complete backend using SQLite with a clean, relational database schema.

This sprint established the system's core structure, data flow, and user experience, serving as a solid foundation for implementing advanced features in future sprints. The system is now stable enough to support the introduction of ride-matching logic, request management, and trip lifecycle features in Sprint 2.

2 Implemented User Stories

2.1 User Story 1: Passenger Registration and Login

This story introduced the basic passenger onboarding flow. The team implemented:

- A passenger registration form with validation for duplicate emails and phone numbers.
- Secure password hashing and a unified login page for all user roles.
- Automatic redirection based on user role after login.
- A simple **Passenger Dashboard** where users can request a ride by entering pickup and destination details.

These features establish the main entry point for passengers in the system.

2.2 User Story 4: Driver Registration and Verification

This story focused on allowing drivers to register and enabling admins to verify them. The implementation includes:

- A full driver registration form with file uploads (ID, license, vehicle documents).
- A linked user–driver database model.
- An **Admin Dashboard** where admins can view pending drivers and approve or reject them.
- Clear status indicators for pending, approved, and rejected drivers.

The flow ensures only verified drivers become active on the platform.

2.3 User Story 5: Driver Online/Offline Status

Approved drivers can manage their availability. This story delivered:

- A **Driver Dashboard** showing personal and verification details.
- An online/offline toggle that updates availability in the database.
- Access control that prevents unapproved or unauthorized users from interacting with driver features.

This completes the basic driver-side functionality required for future ride-matching.

3 Architecture and Implementation Highlights

3.1 Technology Stack

The system was built using a lightweight and easy-to-deploy stack suitable for rapid development:

- **Backend:** Python Flask (routing, session management, authentication)
- **Database:** SQLite (simple relational storage for users and drivers)
- **Frontend:** HTML, CSS, and Jinja2 templating
- **Version Control:** Git and GitHub for team collaboration

This stack allowed the team to quickly produce a functional prototype while maintaining clear separation between logic, templates, and data.

3.2 Database Schema

The database consists of three main tables:

- **users:** stores core user accounts (passengers, drivers, admins)
- **drivers:** stores driver-specific details and uploaded documents
- **driver_status:** tracks each driver's online/offline availability

The schema follows a simple relational structure with foreign keys linking drivers to their corresponding user accounts.

3.3 Key Backend Endpoints

Several important endpoints were created to support the sprint's core functionality:

- `/passenger/register` and `/passenger/login` for passenger onboarding
- `/driver/register` for driver onboarding and document submission
- `/admin/drivers` for viewing and managing driver applications
- `/driver/dashboard` for displaying verification status and toggling availability
- `/passenger/dashboard` for submitting ride requests

These endpoints represent the core flows of the system and enable all user roles to interact with the platform correctly.

4 Sprint Board, Process, and Progress Metrics

4.1 Sprint Board

The team managed Sprint 1 using a Jira Scrum board hosted at:

https://ridehailing.atlassian.net/jira/software/projects/SCRUM/boards/1

The board was organized into three standard Scrum columns:

- **To Do** – User stories and tasks planned for the sprint.
- **In Progress** – Work currently being implemented by team members.
- **Done** – Completed tasks that met the acceptance criteria.

A key configuration detail is that our Jira Scrum board automatically removes completed items from the Active Sprint view once they enter the *Done* state. As a result, the backlog does not display a dedicated "Done" section while the sprint is active, and completed issues cannot be screenshotted directly from the board. Nevertheless, all completed work remains fully visible through Jira's sprint analytics, including the Burndown Chart, Velocity Report, and Issue Navigator.

4.2 Process and Task Distribution

The team followed a lightweight yet effective Scrum workflow during Sprint 1:

- Tasks were selected collaboratively from the refined Sprint 1 backlog.
- Each member moved issues from **To Do** to **In Progress** when development began.
- Continuous collaboration, small handoffs, and frequent GitHub commits ensured synchronized progress.
- Once acceptance criteria were met, tasks were moved to **Done** and the implementation was merged into the main repository.

Progress followed a realistic pattern: features were developed steadily throughout the sprint, and final integration, debugging, and verification were completed near the end of the cycle. Backend logic, session handling, and database design were primarily handled by the team member leading server-side work; driver onboarding and document upload flows were implemented by members focusing on form processing; and HTML structure, dashboard layouts, and UI consistency were achieved through contributions from multiple team members. This distributed effort enabled the team to complete all planned user stories within the sprint timeline.

4.3 Burndown Chart

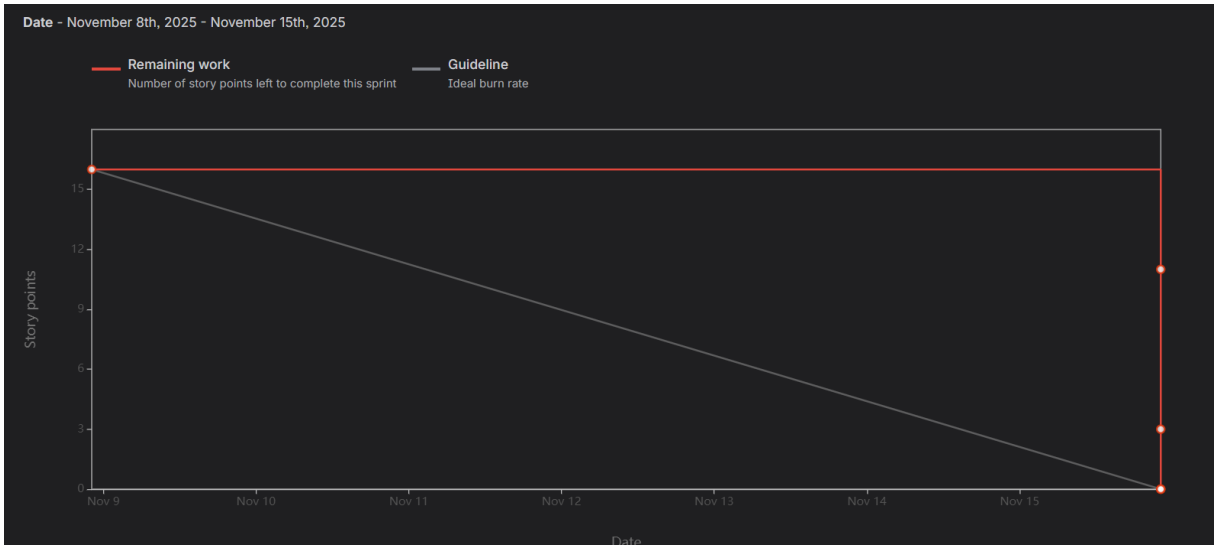

Figure 1: Jira Burndown Chart for Sprint 1. Remaining story points (red) contrasted with the ideal guideline (grey).

Figure 1 shows the progression of remaining work during Sprint 1. The line remains flat for most of the sprint because tasks were in development but not yet transitioned to the *Done* column. As integration and validation were completed near the end of the sprint, all user stories were moved to *Done*, resulting in a single sharp drop to zero. This behavior is consistent with Jira's settings for hiding completed items from the active board and reflects the team's workflow of merging and finalizing tasks toward the close of the sprint.

4.4 Velocity Report

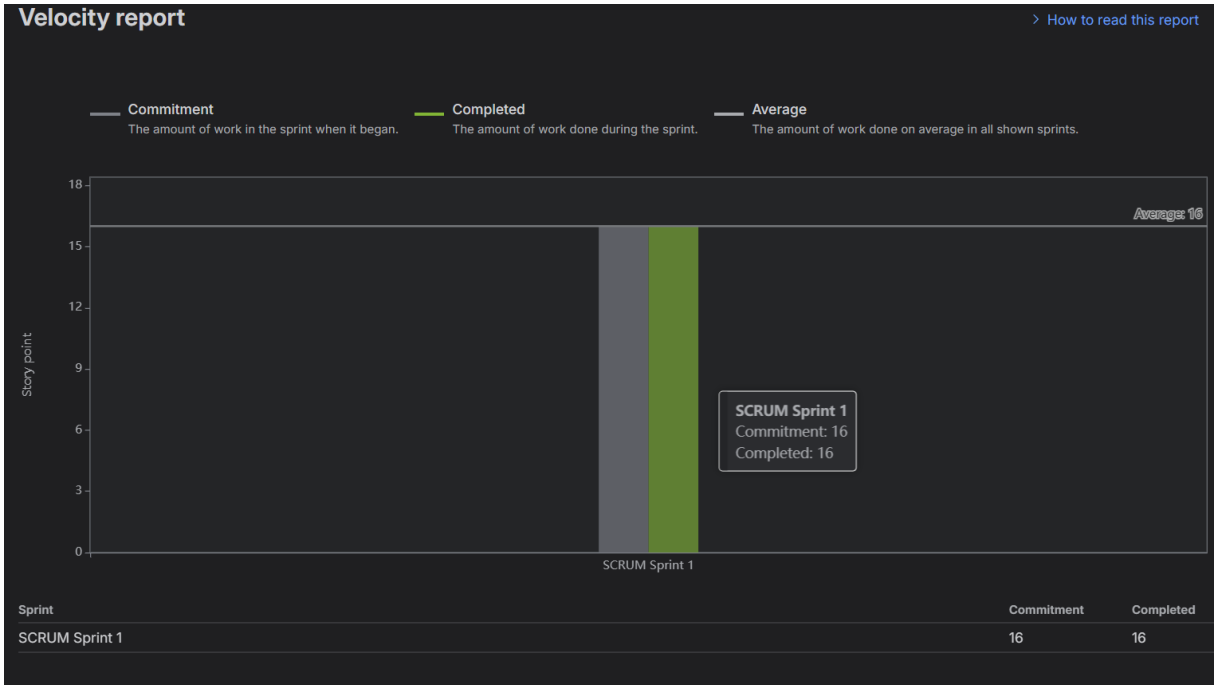

Figure 2: Jira Velocity Report showing commitment and completion for Sprint 1.

As shown in Figure 2, the team committed to **16** story points and successfully completed all **16** by the end of Sprint 1. This indicates accurate sprint planning and demonstrates that no work was carried over to the next sprint. Together with the burndown chart, the velocity report provides quantitative confirmation that User Stories 1, 4, and 5 were completed within the sprint timebox despite Jira's configuration hiding completed tasks from the backlog.

5 Demo and Working Software

5.1 Demo Video Link

A short walkthrough video demonstrating the functionality implemented in Sprint 1 is available at the following link:

Demo.mp4

The demo showcases:

- Passenger registration, login, and dashboard interactions.
- Driver registration with document uploads.
- The admin dashboard for approving or rejecting driver applications.
- The driver dashboard and online/offline availability toggle.

5.2 How to Run the System Locally

The application can be run locally using Python and Flask. The following instructions summarize the setup process as documented in the project repository's README.[1]

Requirements

The application requires the dependencies listed in `requirements.txt`, which include Flask and Werkzeug.

[1] See project documentation at: github.com/3-bhd/ride-hailing-app-team6.

6 Code Evidence

6.1 Repository Link

All source code for Sprint 1 is hosted on GitHub at the following repository:

https://github.com/3-bhd/ride-hailing-app-team6

The repository contains the full implementation of the sprint's user stories, including backend logic, database schema, templates, and static assets.

6.2 Team Commit Contributions

All team members contributed meaningfully to Sprint 1, and each member has at least one commit merged into the repository:

- **Omar Abdelhady (3-bhd)** – Main backend development including authentication, passenger and driver logic, admin approval system, database initialization, and final integration.
- **Janna Osama (Jannaosama)** – Template updates and frontend adjustments.
- **Farida Elanany (faridaelanany)** – Contributions to driver approval workflows and template refinements.
- **George Christo (georgechristo-ctrl)** – Worked on a separate frontend branch and contributed multiple HTML and UI updates before merging into the main branch.
- **Abdelrahman Eid (Abdelrahman-233)** – Driver registration page structure and template-related contributions.

This commit history verifies that each team member participated in the sprint and contributed code to the final deliverable.

6.3 Branch Conventions and Merge Activity

The team followed a simple two-branch workflow suitable for Sprint 1:

- The **main branch** served as the primary development branch where the entire backend and core functionality were implemented.
- A secondary branch was used by **George** for frontend-related work, allowing him to update templates and UI components without interfering with backend development.
- Once stable, George's branch was merged into the main branch, ensuring all final features were integrated before the sprint deadline.

This workflow minimized conflicts and allowed the team to work in parallel while keeping the integration process straightforward. All completed changes were merged into the main branch before the close of Sprint 1.

7 Sprint Retrospective

7.1 What Went Well

The team made strong progress during Sprint 1 and successfully delivered all required user stories. We managed to build a fully functional system structure starting from an empty repository and gradually expanding into a working prototype. The major achievements include:

- Establishing a clear project structure with organized templates, static files, and database helpers.
- Implementing complete authentication flows for passengers, drivers, and admins.
- Creating multiple dashboards (Passenger Dashboard, Driver Dashboard, Admin Dashboard) that dynamically update based on user roles.
- Adding role-based redirection after login, which greatly improved the realism and usability of the system.
- Achieving full UI consistency across all pages, including flash messages, cards, hero sections, and access denied pages.
- Successfully integrating document uploads for drivers and an admin verification workflow.
- Implementing the online/offline status toggle for approved drivers.
- Using Git and GitHub effectively to coordinate contributions and track progress.

Overall, the sprint delivered a stable foundation that the team can confidently build on in future sprints.

7.2 What Did Not Go Well

Although the sprint was successful, several challenges appeared throughout development:

- Multiple routing and endpoint issues occurred at the beginning, especially when extending the base template and adding the logout logic.
- The login flow had to be redesigned several times (from a passenger-only login to a unified login system) to make the experience realistic.
- UI inconsistencies appeared when flash messages or redirects were triggered, requiring additional cleanup for styling and layout.
- Some features initially produced plain white or empty pages due to missing templates or incorrect redirects.
- Role-based access control needed several adjustments before becoming fully stable.
- Certain ideas—such as deploying the system online, adding more advanced dashboards, and implementing ride-matching—were postponed due to scope and time constraints.

These challenges were resolved step by step, but they highlighted the need for clearer planning and template consistency early in the sprint.

7.3 Improvements for Next Sprint

Based on the sprint's experience, the following areas will be improved moving forward:

- Finalize and polish UI components early to avoid layout inconsistencies in later stages.
- Introduce clearer task breakdown so that feature dependencies (e.g., login before dashboards) are completed in the correct order.
- Begin planning for deployment to allow external testing and easier demonstrations.
- Expand the database schema to support rides, locations, pricing, and notifications.
- Consider creating reusable Jinja components for cards, alerts, and layout blocks.

These improvements will help the team deliver more complex features efficiently in the next sprint.

8 Team Reflections & Roles

8.1 Team Roles

To organize the work for Sprint 1, the team adopted lightweight Scrum roles that matched the project's needs and team strengths:

- **Omar Abdelhady** – *Product Owner & Scrum Master*. Facilitated planning, clarified user stories, ensured that acceptance criteria were met, and supported the team in coordinating tasks. Also contributed as a developer on several backend components.
- **George Christo** – *UI/Frontend Specialist*. Focused on HTML templates, layout adjustments, and improving the overall user interface.
- **Janna Osama** – *Developer*. Contributed to template refinements and supported frontend integration.
- **Farida Elanany** – *Developer*. Worked on parts of the admin verification flow and contributed to template updates.
- **Abdelrahman Eid** – *Developer*. Implemented the driver registration interface and supported template structure changes.

8.2 Reflection on Collaboration

The team collaborated effectively throughout Sprint 1, dividing tasks based on individual comfort levels and strengths. Developers worked on both backend and frontend elements, while the UI specialist handled the interface structure. Coordination took place through Jira and GitHub, with frequent communication to avoid conflicts and ensure smooth integration.

Although responsibilities varied slightly based on experience, every member contributed code that was merged into the repository, resulting in an even and balanced team effort across the sprint.

8.3 Adjustments Planned for Sprint 2

Looking ahead to Sprint 2, the team plans to:

- Maintain the same role structure to preserve clarity and efficiency.
- Increase shared work sessions so that members can collaborate more closely on features.
- Provide additional support for UI and templating tasks to spread workload more evenly.
- Continue using Jira more actively to visualize progress throughout the sprint.

These adjustments aim to enhance collaboration and ensure an even smoother workflow in the next sprint.

9 Burndown / Progress Summary

To summarize Sprint 1 progress, the team committed to **16 story points** and successfully completed all **16 story points** by the end of the sprint. This is reflected in the Velocity Chart, which shows a full match between planned and completed work.

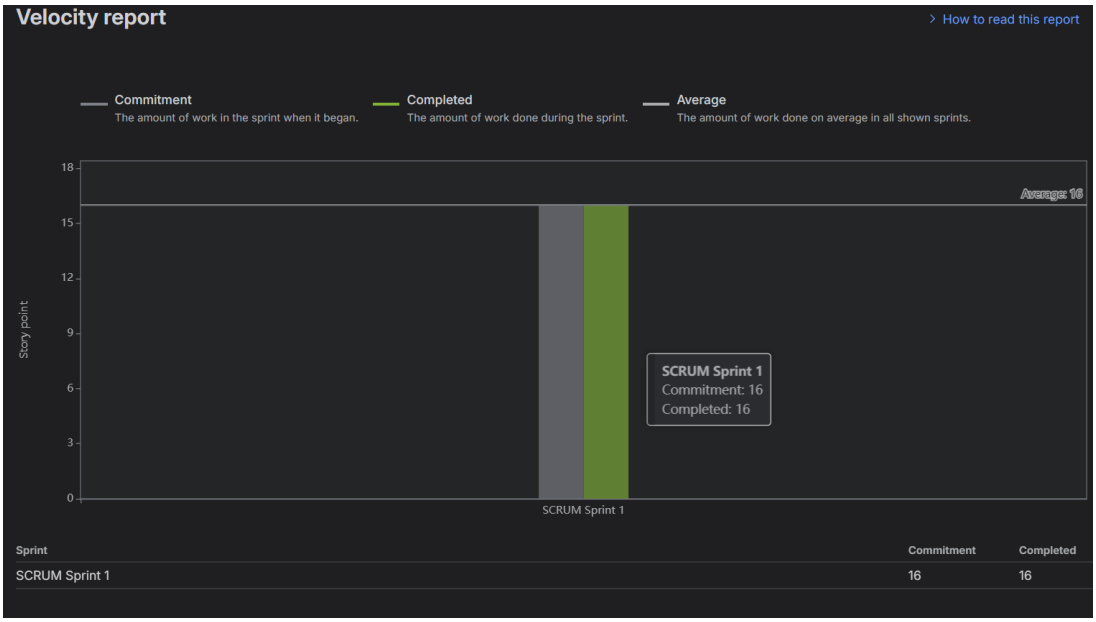

Figure 3: Sprint 1 Progress: Story Points Planned vs. Completed.

This confirms that all selected user stories for Sprint 1 were completed within the sprint timebox, with no spillover to the following sprint.

A Appendix: Commit History Evidence

This appendix provides supporting evidence of the team's contributions during Sprint 1. All commits were merged into the shared GitHub repository, demonstrating active participation from each team member.

Repository Link

https://github.com/3-bhd/ride-hailing-app-team6

Commit Log Summary

The following commits (shown in Figures 4, 5, 6) and illustrate individual contributions across Sprint 1. Each team member has at least one commit merged into the repository, fulfilling the sprint requirement for shared collaboration.

Figure 4: Excerpt of GitHub commit history showing team contributions (Part 1).

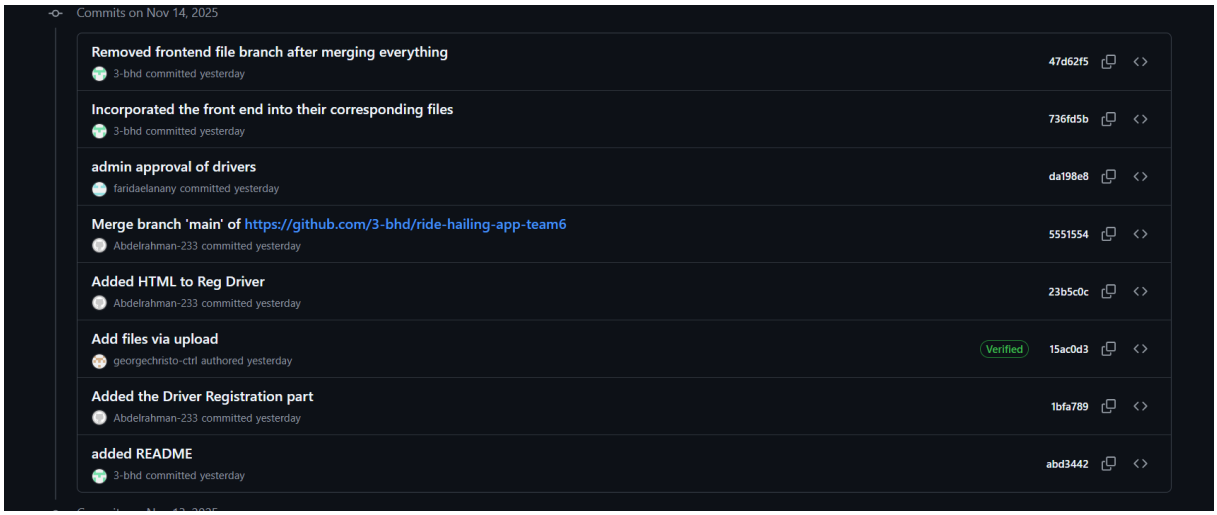

Figure 5: Excerpt of GitHub commit history showing team contributions (Part 2).

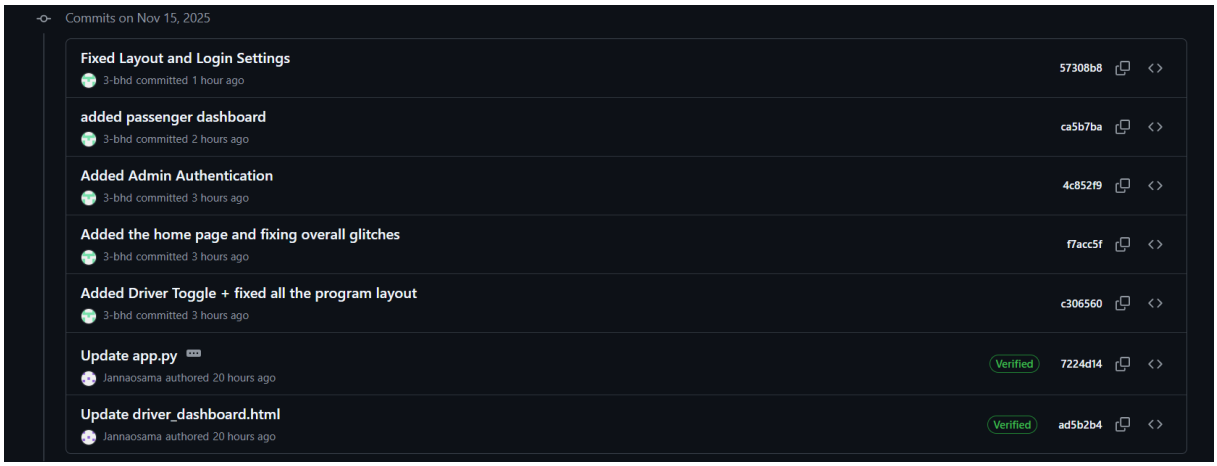

Figure 6: Excerpt of GitHub commit history showing team contributions (Part 3).

Notes on Evidence

- The commit history confirms that all team members contributed code during Sprint 1.
- Contributions include backend logic, frontend templates, UI updates, and document handling.
- Additional commits may be viewed directly in the repository for full transparency.