CS 4850

Fall 2024

SP-06-Red-Time Mileage Tracker with GPS

Parth Patel, Ryan Thompson, Lloyd Anderson, Humberto Aguilar-Sanchez, David Lowe

Professor Sharon Perry

30 November 2024

Website Link: https://tmt.thomp.site

GitHub Link: https://github.com/mileagettracker

Total lines of code: 6,952

Number of components/tools: 5

Total man hours: 125

Table of Contents

1.0 Introduction

1.1 Overview

The Time Mileage Tracker with GPS is a mobile app enabling employers to ensure employees are efficiently managing time and mileage. Employers can access data on employees' activities to verify time and mileage reports. The app aims to improve productivity, reduce user input errors, and provide accurate records for employers.

1.2 Project Goals

To develop a web application that employers can use to track their employees' mileage to assist in optimizing driver routes, reducing spending, and reimbursing drivers.

1.3 Definitions and Acronyms

- MB- megabyte: unit of memory storage.
- RAM- random access memory: the computer hardware that provides temporary storage to the computer and its programs.
- API- Application Programming Interface: an interface allowing two or more computer programs or components to communicate with each other.
- GPS- Global Positioning System: a satellite-based radio navigation system that most modern mobile devices are manufactured with.
- EU- The abbreviation for the continent of Europe and for the European Union.
- GDPR- General Data Protection Regulation: a European Union regulation on information privacy in the European Union and the European Economic Area.
- Email- Electronic Mail: Denotes email in the general sense such as Gmail, Outlook, Yahoo, and any other email service available.
- React - A JavaScript library that helps developers create graphical interfaces for web applications and native platforms.
- Django - A free and open-source web framework that runs on a web server, and uses the Python programming language.

- ORM - Object-Relational Mapper, which is a software tool that translates data representations between databases and object-oriented programming languages.
- Marvel App - An application that design teams use to prototype, test, and handoff designs.
- SQLite - a database engine written in the C programming language.
- Apache - free and open-source cross-platform web server software
- IOS- iPhone Operating System
- Wi-Fi- Wireless Fidelity
- SSL- Secure Sockets Layer
- HTTPS- Hypertext Transfer Protocol Secure
- JSON- JavaScript Object Notation
- DB- Database
- IDE- Integrated Development Environment

1.4 Assumptions

The users will be provided with company devices that meet the requirements of the application

2.0 Design Constraints

The application supports different user roles which have different access levels and proper user authentication.

2.1 Environment

Internet connection of WI-FI or cellular data is required, as it is a web-based application that uses GPS

2.2 System

Android device with version 8.0 or later

IOS device with version 16.0 or later

3.0 Functional Requirements

3.1 Dashboard Interface

User Login

Sign in if the account already exists

Or create an Account using an email address and password

Save the account to the SQLite database

Account recovery

- If the user does not remember their password, they can provide their email address and will be redirected to a password reset page.

Dashboard Overview

Navigation

-After selecting ”Click here” under “Navigation”, the user enters their destination, with their current location already detected. Then the user will click ‘navigate’, in which they will be taken to Google Maps.

-The option to return to the dashboard is listed here on the left side of the screen, before the user selects the “Navigate” button.

User Profile

-Here, the user may select “View History” in order to view their route history. The information of start location, end location, distance, duration, and timestamp are given here.

-To return to the dashboard, the user may select “Dashboard” on the right side of the Route History pane.

Employee Reports

-Here, the user may select “view reports”, upon which they will be prompted to enter their administrator login, under the regular login page.

-Upon a successful administrator login, the administrator will be brought back to the dashboard, in which they will select “view reports” once more.

-Next, the Employee Reports pane appears, in which all of the user mileage data is available. Included in this pane are the employee username, start location, end location, distance, and duration. There are also buttons to “edit” and “export PDF”.

-In the event that the administrator needs to edit a report, this button can be used to modify any of those values that are given in the Employee Reports pane. They may select “back” to return, otherwise the modification may be saved, and then the admin will return to the previous page.

About

-Here, the user may select “About” in order to view the project overview, project deliverables, and presentation.

-To return to the dashboard, the user may select “back to Dashboard” on the right side of the Project Overview pane.

.

3.2 Reports on Collected Data

Admin Dashboard

Employee Reports

– Shows where employees traveled, how many miles they traveled, and how long it took them.

– Allow the administrator to export the report as a PDF

Modify Report

– Gives administrators the ability to manually override employee reports data.

3.3 GPS Tracking

Google Maps API

-Record traveled distance, start/end location, and start/end time

-Save to the database and be accessible to administrators.

3.4 SQLite Database

auth_user Table

Purpose: Stores information about each user, including login credentials, account details, and their association with a group.

Fields:

Username: Identifies the user uniquely within the system.

Password: Securely stores user passwords (hashed).

Administrator Status: Indicates whether the user has administrative privileges.

Associated Group: Links the user to a specific group (auth_group) for role-based access or categorization.

auth_group Table

Purpose: Defines groups that users can belong to and manages ownership and categorization of users. Useful for implementing group-based permissions or functionality.

Fields:

Group ID: Unique identifier for each group.

Group Name: A descriptive name for the group like a company name

Owner: Indicates the owner or administrator of the group, linking to the auth_user table.

Creation Date: Records when the group was created for tracking purposes.

time_mileage_tracker_routelog Table

Purpose: Tracks information about user travel activities, including distance, duration, and locations.

Fields:

Numerical Data:

Distance Traveled: Total number of miles/kilometers covered in a trip.

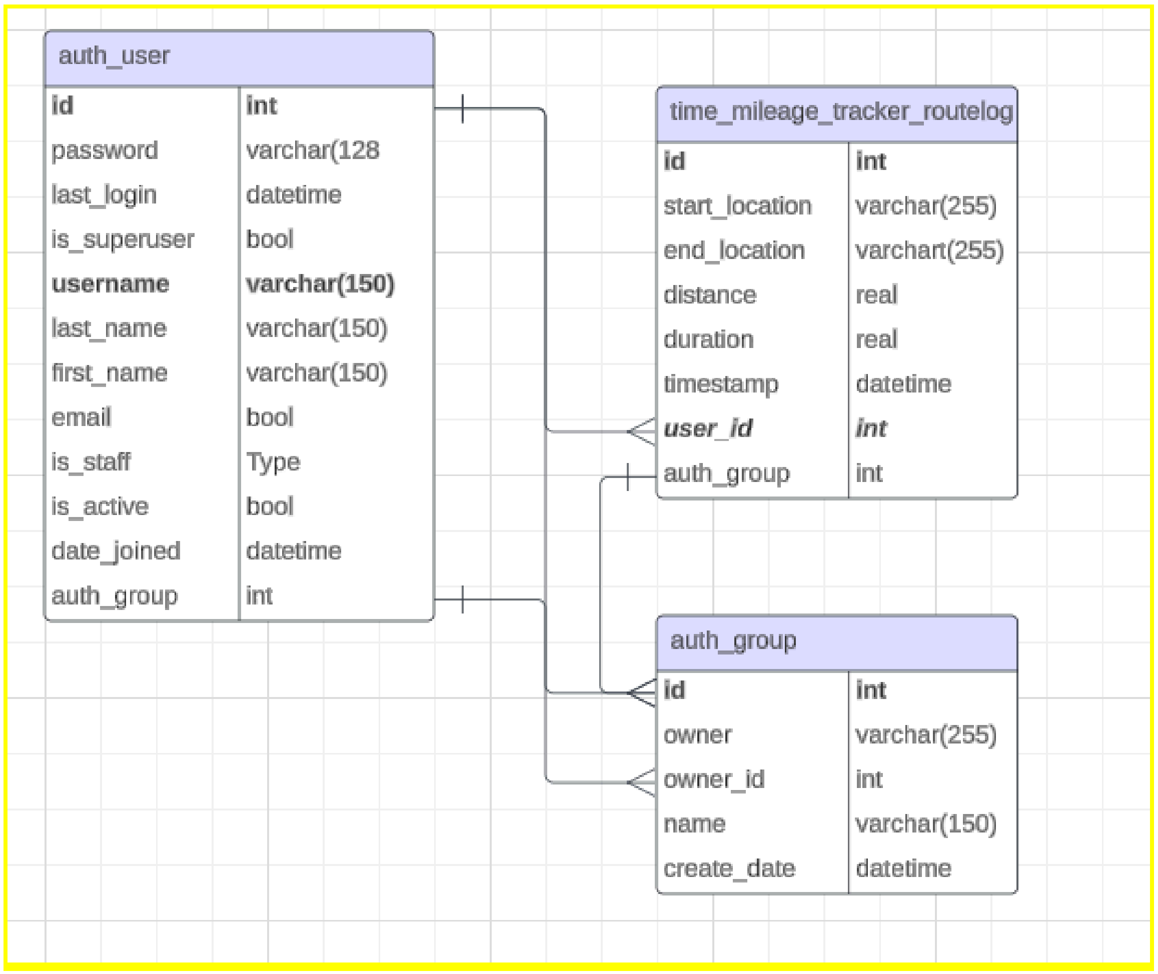
Duration: Time taken for the trip, recorded in hours/minutes.

Non-Numerical Data:

Start Location: Starting point of the trip.

End Location: Destination of the trip.

Timestamp: Date and time the trip was logged.



4.0 Non-Functional Requirements

4.1 Security

SSL and HTTPS will be used to maintain security when transferring the GPS data to the admin application page

4.2 Capacity

Data will be sent over the internet to the external database to be stored for later viewing

5.0 External Interface Requirements

5.1 Software Interface Requirements

Data Formats: will include JSON to store and maintain the session of a user tracking history that JSON will be sent to the database for storage

5.2 Communication Interface Requirements

Web Sockets: Will be used to transfer and transmit data to and from the device within the web application

6. Design

6.1 Assumptions and Dependencies

- Software that can run the latest version of Google Chrome, such as iOS 16 and Android 8
- Assumed to be compatible with all main operating systems, as it is web-based
- Device is assumed to have the ability to use location services
- Device is assumed to be a mobile device such as a smartphone so that movement may be traced
- User must have a functioning email address
- User is above the age of 4

6.2 General Constraints

- Device must have a user interface
- Data storage is required to run admin reports
- Device must have proper power (battery life)
- Device must have 300 MB of RAM available
- Strong GPS signal is required to maintain proper accuracy when tracking a device.
- GDPR Compliance would be required if the application is to be used in the EU, storing of personal data and information must be done securely and properly
- Application will use Maps APIs and the device must be able to use and handle these APIs allowing for a proper exchange of information

7.0 Development

7.1 Development Methods

We used the waterfall development method by first establishing our requirements where we define the problem and ways we can solve the problem. We created a design based on these requirements via a mockup in the Marvel App that outlines the structure, subsystems, and interfaces. We then converted the design that we created with Marvel into source code with the components of the system.

7.2 Architectural Strategies

- Programming Language and Framework
  - Python is the main programming language and uses the microframework called “Django” which allows for Python-driven apps and an HTML front end. Python is a modern and robust language with almost universal support. Using Django allows for us to have all intensive processes to be handled on the server backend not on the users device.
  - The choice of Django as the primary framework was driven by its robust ecosystem and built-in features such as the ORM, which simplifies database interactions, and its strong community support.
  - Django's "batteries-included" philosophy allows for rapid development and reduces the need for third-party dependencies, which aligns with maintaining a streamlined and manageable codebase.
- User Interface Paradigm
  - A responsive web design was adopted for the user interface, utilizing modern frontend frameworks like React.
  - This approach ensures that the application is accessible across a variety of devices, catering to drivers and administrators who may access the system on the go. The use of React also allows for a dynamic, real-time user experience.
- Data Management

    - SQLite was chosen as the database engine to store user and admin login data and mileage report data, as it is a built-in function of Django and is well-integrated therein.\
- GPS Tracking
    - For GPS Tracking and routes, we went with the Google Maps API to allow the user to navigate and track their miles. This data is stored on the host machine and is used for further reporting to an admin. But the primary driving app for tracking is Google Maps with its robust routing algorithms and consistency of tracking and location.

7.3 System Architecture

- The Django framework is hosted on an Apache web server
- Presentation Layer- Google Chrome
- Session Layer- Google Maps API and sockets to communicate GPS data
- Network Layer- IP
- Data Link Layer- Wi-Fi to transmit data
- Physical Layer- Personal Computer/Mobile Device

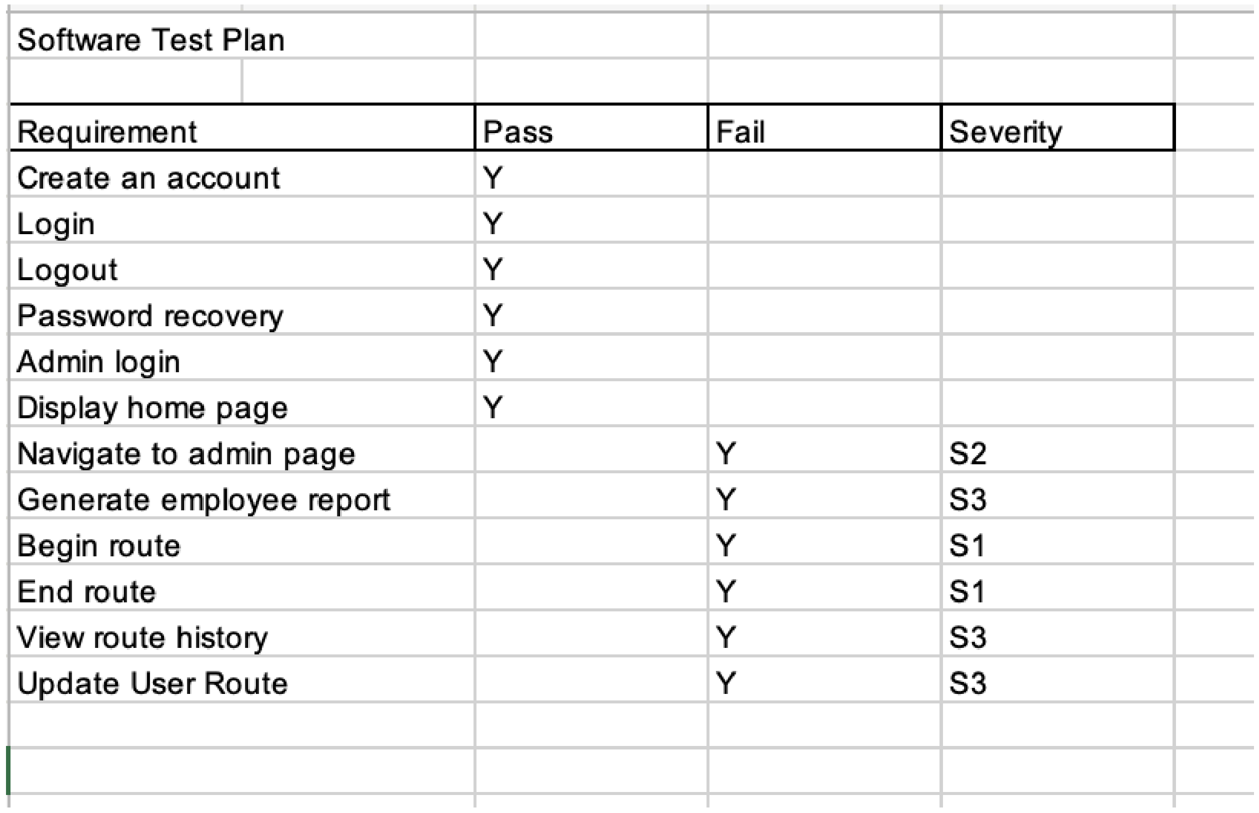
8.0 Testing

- ***8.1 Test Plan***

The test plan phase includes testing all the parts of the application in multiple different ways.

- o Unit Testing: Testing certain parts of the application such as the mileage tracking, GPS, user login, etc. to ensure accuracy.
- o Integration Testing: Testing how different parts of the application such as the GPS and database logging work together to ensure accuracy.
- o System Testing: Testing the full application on different web browsers and different devices to ensure accuracy.
- o See below for an image of the software test plan, in which each main requirement has been tested by all group members and marked as having either passed or failed and given a security level of S1, S2, or

S3, with S3 being the most severe and S1 being the least severe. “Y” indicates “Yes” here. Note that this table was created using Microsoft Excel.



Software Test Plan

Requirement	Pass	Fail	Severity
Create an account	Y		
Login	Y		
Logout	Y		
Password recovery	Y		
Admin login	Y		
Display home page	Y		
Navigate to admin page		Y	S2
Generate employee report		Y	S3
Begin route		Y	S1
End route		Y	S1
View route history		Y	S3
Update User Route		Y	S3

- ***8.2 Test Report***

After we complete the test plan phase, we move on to the test report phase which involves testing the application for bugs and its performance metrics.

  - Bug Reports: Making reports of the bugs we have encountered and how to solve them.
  - Performance Metrics: Making reports of the performance of the application including load times and responsiveness with various use case scenarios.

9.0 Version Control

- ***9.1 Git***

We used Git for version control with the source code hosted on GitHub because of its robust support for branching, merging, and rollback.

- Branching: Allows us to work on multiple parts of our application without changing the main application.
- Merging: Allows us to combine our code changes with another branch of code.
- Rollback: Allows us to revert our application to a previous state in case of new issues that have just arisen.

10.0 Conclusion

- We have created a Time Mileage Tracker with GPS that enables employers to ensure that employees are efficiently managing time and mileage in which employers can access data on employees' activities to verify time and mileage reports.
- The future scope of this application will involve monetization in order to keep it free to our users. That is, there will be in-application advertisements that will not affect the performance of the application or the administrator's ability to run employee reports. This will most likely involve Google advertisements, but we are open to other tools and resources that will allow us to achieve this goal. Monetization was not a priority for this version of our web application, as we first aimed to ensure a user-friendly experience to be expanded upon at a later time. This application is currently free to access and will remain so in the future.
- Our team is committed to ensuring a satisfactory experience and encourages feedback and the reporting of any issues or errors that may be encountered. Please do not hesitate to contact any of us at our email addresses listed on the cover page of this document.

APPENDICES

Appendix A- Project Plan

Time Mileage Tracker - Project Plan

The Time Mileage Tracker with GPS is a mobile app that enables all employers to ensure employees are efficiently managing time and mileage. Employers can access data on employees' activities to verify time and mileage reports. The app aims to improve productivity, reduce user input errors, and provide accurate records for employers.

2.0 Project website

www.mileagetracker.com

Deliverables

- ☐ Project plan draft
- ☐ Requirements
- ☐ Design
- ☐ Prototype design
- ☐ Final report and presentation
- ☐ Final product
- ☐ (optional use tutorial)

Milestone Events

- ☐ Planning documentation complete - September 1, 2024
- ☐ Concept design complete - September 15, 2024
- ☐ Prototype Complete - October 1, 2024
- ☐ Prototype presentation - October 15, 2024
- ☐ C Day - November 19, 2024
- ☐ Application fully developed - December 2, 2024
- ☐ Final report - December 2, 2024

Meeting Schedule Date/Time

Bi-weekly status meetings on Thursdays from 4:15 pm to 4:30 pm.

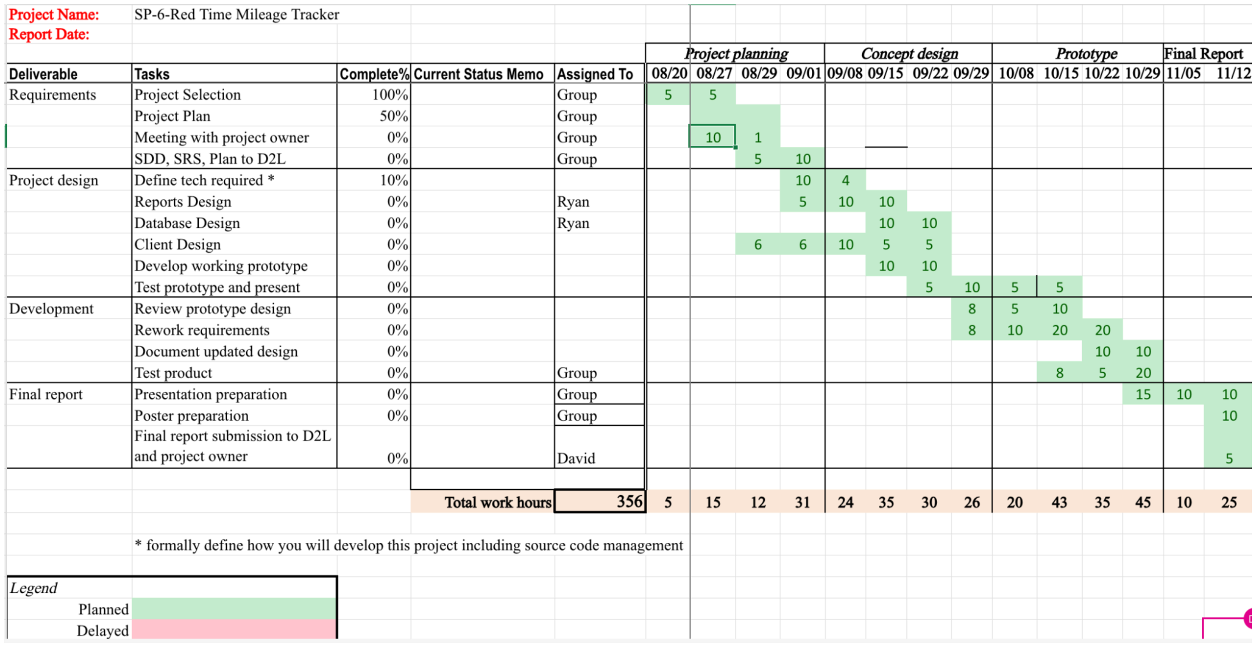
Collaboration and Communication Plan

Weekly Discord meetings on Mondays and Wednesdays at 4:00pm.

Text message communication in group chat.

Project Schedule and Task Planning

See the Project Work Plan (Gantt chart) file attached.



Project Name: SP-6-Red Time Mileage Tracker

Report Date:

					Project planning				Concept design				Prototype				Final Report	
Deliverable	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**	**08/20**	**08/27**	**08/29**	**09/01**	**09/08**	**09/15**	**09/22**	**09/29**	**10/08**	**10/15**	**10/22**	**10/29**	**11/05**	**11/12**
Requirements	Project Selection	100%		Group	5	5												
	Project Plan	50%		Group														
	Meeting with project owner	0%		Group		10	1											
	SDD, SRS, Plan to D2L	0%		Group			5	10										
Project design	Define tech required *	10%						10	4									
	Reports Design	0%		Ryan				5	10	10								
	Database Design	0%		Ryan						10	10							
	Client Design	0%					6	6	10	5	5							
	Develop working prototype	0%								10	10							
	Test prototype and present	0%									5	10	5	5				
Development	Review prototype design	0%										8	5	10				
	Rework requirements	0%										8	10	20	20			
	Document updated design	0%													10	10		
	Test product	0%		Group										8	5	20		
Final report	Presentation preparation	0%		Group												15	10	10
	Poster preparation	0%		Group														10
	Final report submission to D2L and project owner	0%		David														5
			Total work hours	356	5	15	12	31	24	35	30	26	20	43	35	45	10	25

* formally define how you will develop this project including source code management

Legend	
Planned	🟩
Delayed	🟥

Version Control Plan

Deploy each milestone onto GitHub and update along the way.

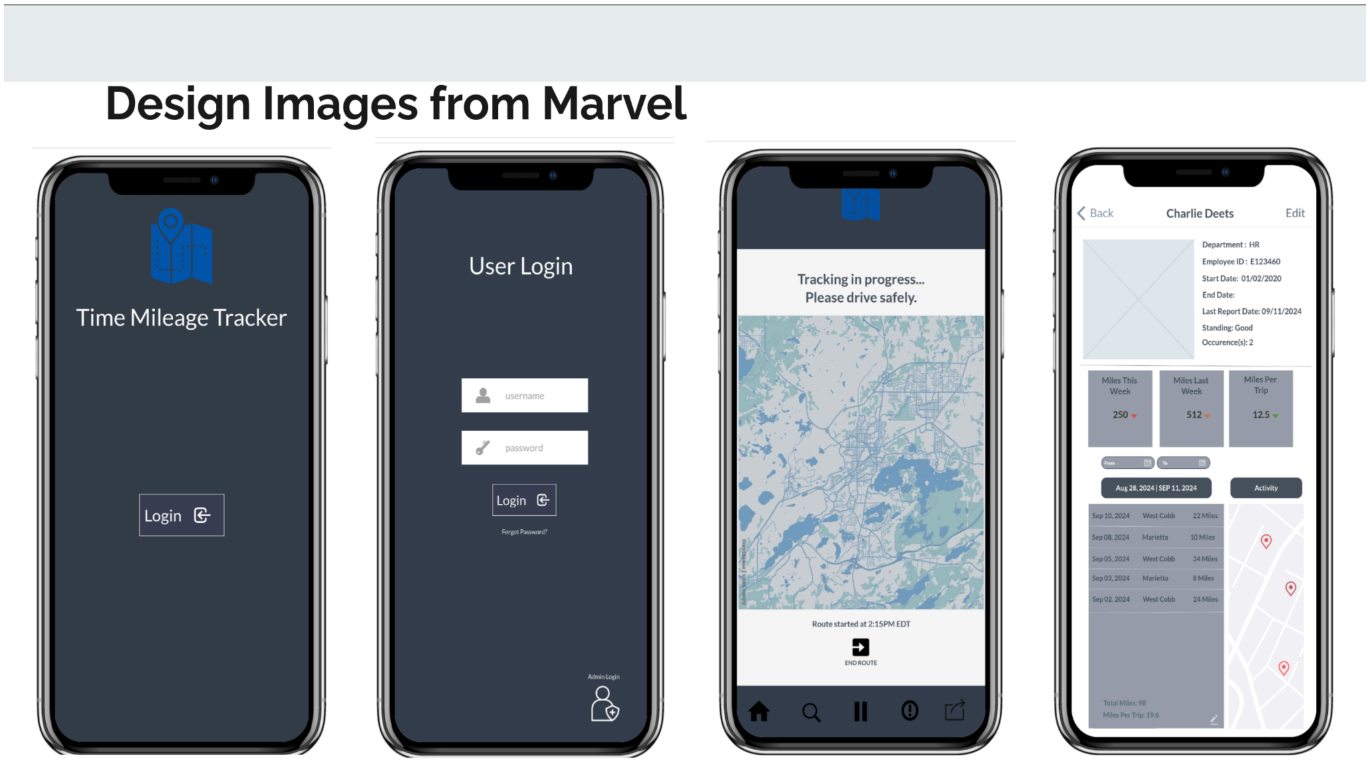
Tutorial Plan

Record a demo video that demonstrates how to use the application for first time users.

Appendix B- Application Design Mockup

In order to visualize our application and avoid missing any key design features during development, our team created an interactive design using the Marvel app.

Marvel, accessible at marvelapp.com, is completely free to use and allowed us to bring our vision for our application to life. Below are some sample user screens from our design mockup:


Design Images from Marvel

Department : HR
Employee ID : E123460
Start Date: 01/02/2020
End Date:
Last Report Date: 09/11/2024
Standing: Good
Occurence(s): 2

Miles This Week	Miles Last Week	Miles Per Trip
250	512	12.5

Aug 28, 2024 | SEP 11, 2024

Sep 10, 2024	West Cobb	22 Miles
Sep 08, 2024	Marietta	10 Miles
Sep 05, 2024	West Cobb	34 Miles
Sep 03, 2024	Marietta	8 Miles
Sep 02, 2024	West Cobb	24 Miles

Total Miles: 98
Miles Per Trip: 19.6

Featured in these user screens are the login screen, including a ‘forgot password’ button, the ‘tracking in progress’ screen, and an example screen of the total mileage for a specified date. This design mockup was purposely tailored to a mobile device, as this is the most probable device that we expect our application to be run on, but almost any device with internet capability and location services will suffice.

Appendix C- Web Application Tutorial

How to set up this web application – steps:

Install latest Django (5.1.3), Python (3.12), and PyCharm (2024.2.4) versions to your device.

Create Django project in PyCharm IDE

Deploy Django development server

- Run the command in PyCharm: python manage.py runserver

Configure 'settings.py' in Django framework

- Change DEBUG feature to 'True'
- Enter the ALLOWED_HOSTS
- Generate a SECRET_KEY

Set up initial files for web view

- Create a view in 'views.py'
- Define a URL configuration in 'urls.py' to access the view in the browser

Create a templates directory in the Django project folder

- Add base.html, which is a template that serves as a base or layout for other templates (i.e. Dashboard, Login, and Signup pages).
- The views.py and urls.py files work together to handle requests and render the appropriate templates

Add Django's built-in SQLite database to store user data

Appendix D- Marvel App Mini-Tutorial

We used Marvel to create a design mockup for our web application because of its dynamic, real-time user experience.

Steps:

Create a Project

Add components: Adding interactive elements such as text, boxes, logos, buttons, menus, etc.

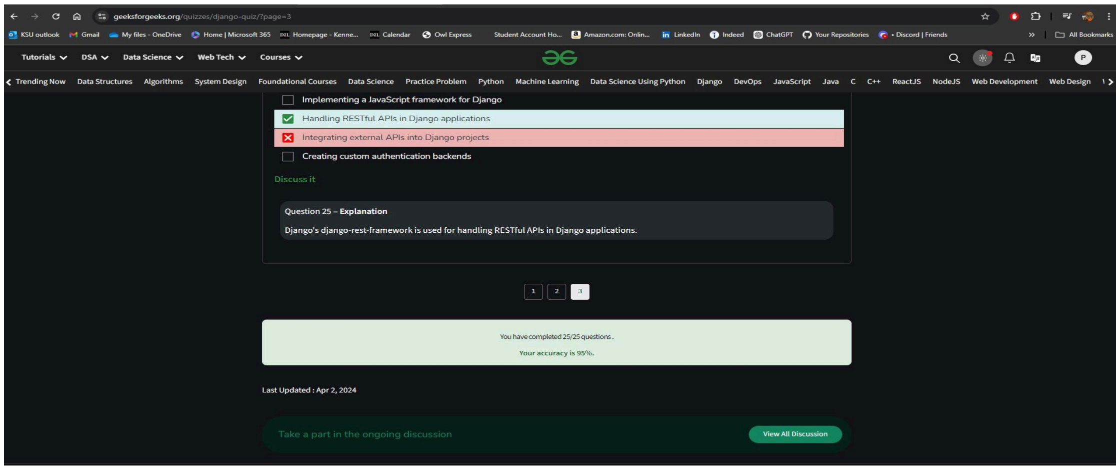
Link the screens together so the clicking each button takes you to the correct screens. This simulates how a user would interact with the application.

Testing: Involves testing for bugs from all units of the application.

Appendix E- Django Tutorial Completion Proof

As not everyone on our team had used Django before this project, every member of our team was required to complete a tutorial before beginning development. We decided to use a simple, beginner-friendly Django tutorial available for free at this link: GeeksForGeeks. A 25-question quiz followed the topics included in the tutorial, of which each team member received a passing score and took a screenshot of their results for proof. Below are these screenshots for each team member:

Parth:



☐ Implementing a JavaScript framework for Django

✓ Handling RESTful APIs in Django applications

✗ Integrating external APIs into Django projects

☐ Creating custom authentication backends

Discuss it

Question 25 – **Explanation**

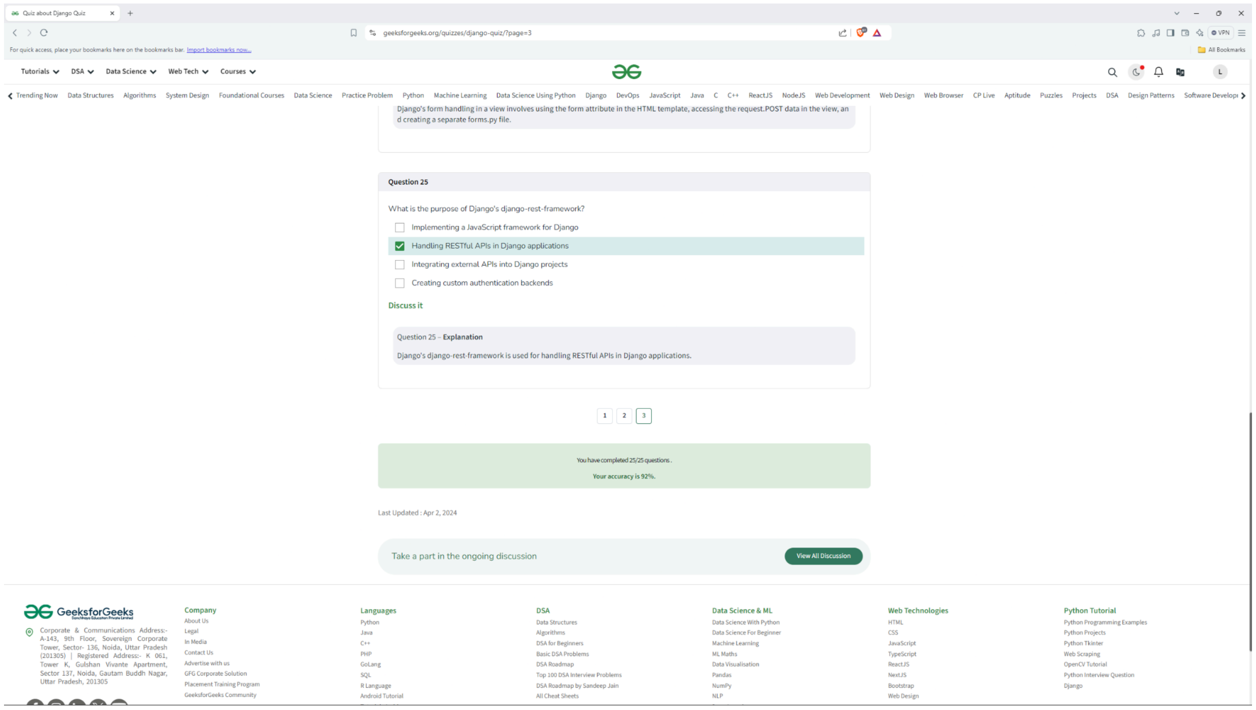
Django's django-rest-framework is used for handling RESTful APIs in Django applications.

You have completed 25/25 questions .

Your accuracy is 95%.

Last Updated : Apr 2, 2024

Lloyd:



geeksforgeeks.org/quizzes/django-quiz/?page=3

Django's form handling in a view involves using the form attribute in the HTML template, accessing the request.POST data in the view, an
d creating a separate forms.py file.

Question 25

What is the purpose of Django's django-rest-framework?

- [ ] Implementing a JavaScript framework for Django
- [✓] Handling RESTful APIs in Django applications
- [ ] Integrating external APIs into Django projects
- [ ] Creating custom authentication backends

Discuss it

Question 25 – **Explanation**

Django's django-rest-framework is used for handling RESTful APIs in Django applications.

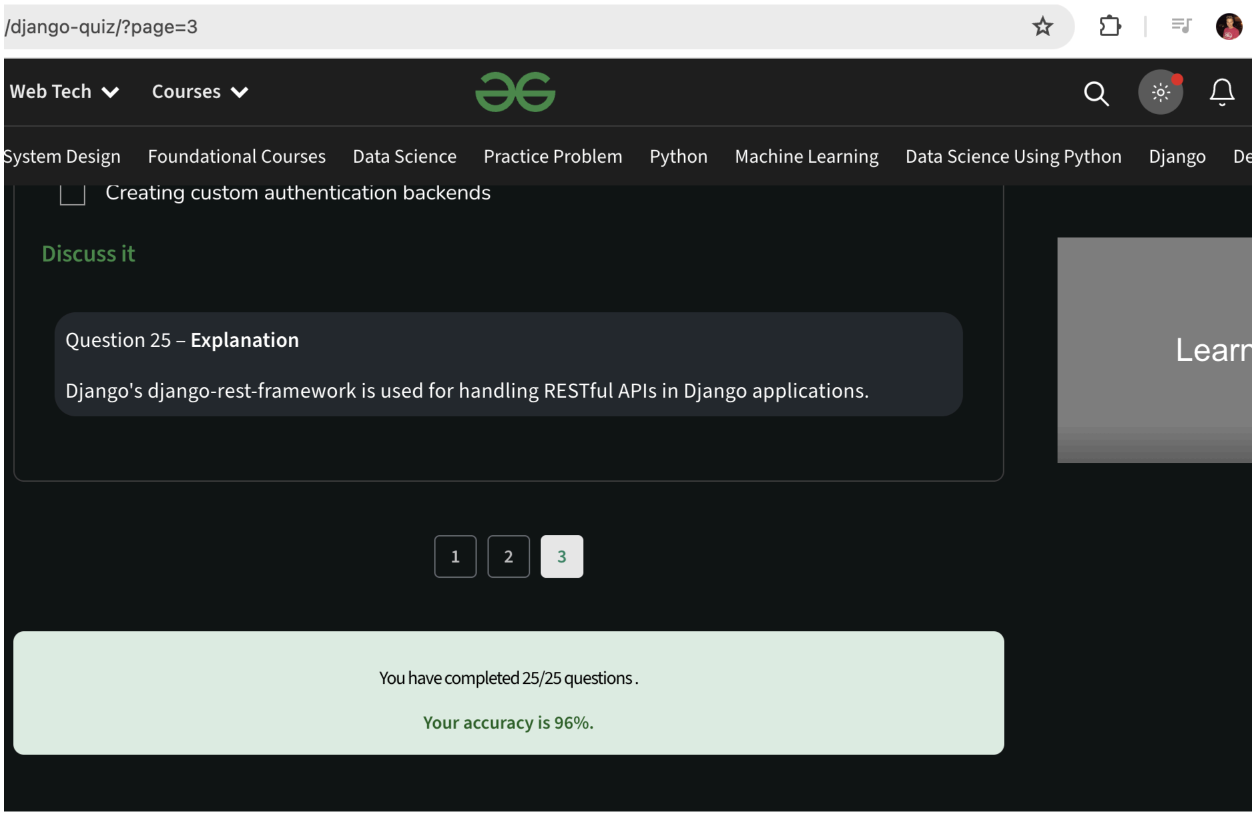
You have completed 25/25 questions .

Your accuracy is 92%.

Last Updated : Apr 2, 2024

Take a part in the ongoing discussion

David:



/django-quiz/?page=3

- [ ] Creating custom authentication backends

Discuss it

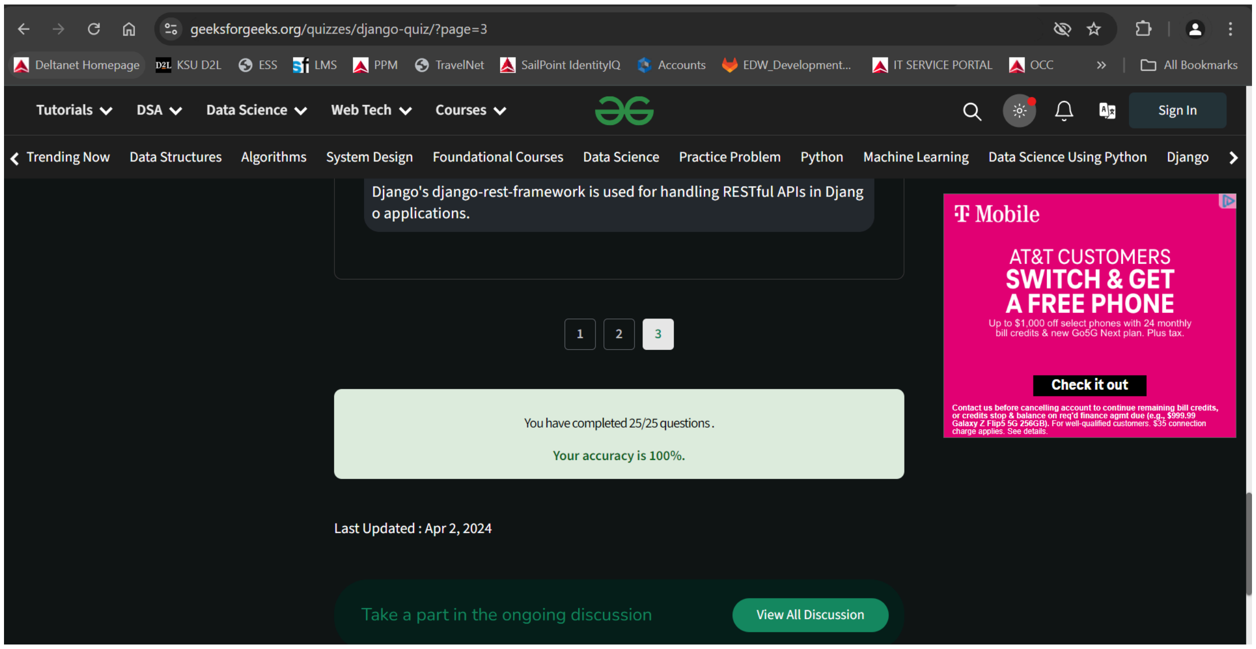
Question 25 – **Explanation**

Django's django-rest-framework is used for handling RESTful APIs in Django applications.

You have completed 25/25 questions .

Your accuracy is 96%.

Ryan:

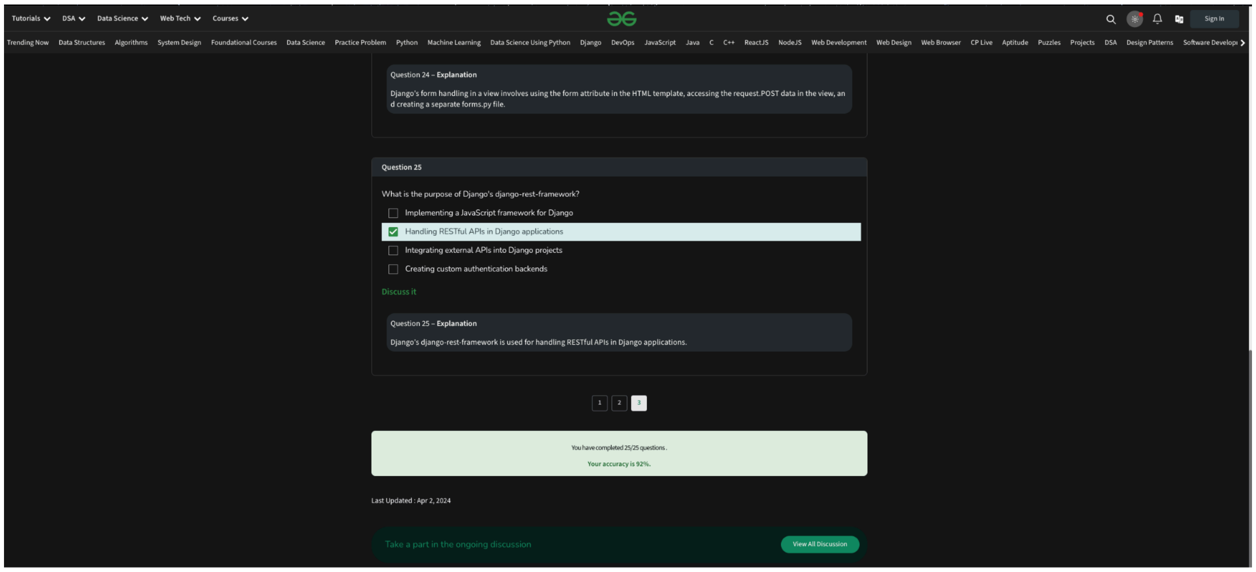

Django's django-rest-framework is used for handling RESTful APIs in Django applications.

You have completed 25/25 questions .

Your accuracy is 100%.

Last Updated : Apr 2, 2024

Humberto:


Question 24 – **Explanation**

Django's form handling in a view involves using the form attribute in the HTML template, accessing the request.POST data in the view, and creating a separate forms.py file.

Question 25

What is the purpose of Django's django-rest-framework?

- [ ] Implementing a JavaScript framework for Django
- [✓] Handling RESTful APIs in Django applications
- [ ] Integrating external APIs into Django projects
- [ ] Creating custom authentication backends

Question 25 – **Explanation**

Django's django-rest-framework is used for handling RESTful APIs in Django applications.

You have completed 25/25 questions .

Your accuracy is 92%.

Last Updated : Apr 2, 2024