Legend (Color Code of Text)	
Compulsory	Can be done in any order, but must be covered during training period
Theoretical Concepts Only	Can be done in any order, but theoretical concepts must be discussed during training period
Optional	Can be done in any order. It is optional to discuss and implement these concepts

	Topics					Tasks / Assignments	Comments
Django	Server and Client Side Concepts. Http methods (GET, POST, PATCH etc).	Python http server (No frameworks), Request & Response Object. Headers	Flask (Routing, Views, Middelware (Concept only), HTML Views, Basic Error Handling, Request data, App Context)	Small & Large Application structure FLASK		Task: Update weather man application to serve data using Flask Application with multiple routes. Flask application will maintain in memory records only. Task B: Add functionality to existing FLASK application of keeping track of users. User data will be saved in simple JSON or Text File for presistiency.	
	Django URLs and Views Basics. Advanced Django Views (Class Based Views)	Django Models (Designing Models, Properties of Models), Migrations, Writing basic Django ORM Queries (create, delete, all, get_or_create, update_or_create, conditional expresssions),	Django Forms, Model Forms, Advanced Django Forms with Default/Initial Values	User Authentication (Login and Logout) Exceptions Handling, Writing Custom Exceptions. Django Decorators	Django Templating Engine. Extending Django templates. Writing Custom Template Tags in Django	TASK: Django Application with Django Form, Authentication (Signup, Login, Logout, Edit Profile), Using Custom Django Templates Proper Exception Handlings	

		filter vs get.					
	Types of Databases & Differences	SQL (CREATE, DELETE, UPDATE, GROUP BY, HAVING, KEYS, JOINS, RELATIONS, TABLES, VIEWS, COLUMN TYPES)	No SQL db types (Key Value, Document, Column Graph). Designing Schema (Access patterns, Partition Keys, Sort Keys)			Task: Implement NoSQL based Document store api for fetching user record	Optional Can be done in any sequence or after advance django topics as well
	Advanced Models (Designing Custom User Model, Inheritance) Squashing migrations, Reversing migrations	Overriding save method of Models. Post/Pre Save i.e., Signals. View decorators, File Handling, Middlewares (Custom Authentications)	Django Model Object Managers, Writing Custom Object Managers Model Level Validations	Django Advanced Queries, bulk_create() bulk_update() select_related() prefetch_related() RawQuery, F(), Aggregate, Annotate, Subquery, Atomic transactions	Django Management Commands Data Migrations Django Admin (model registrations) Custom displays Custom filters on list displays Customizing search filters	TASK: A Complete Django Application with Custom User Model, Custom Authentication Middleware, Auto Generated Fields in the User Model that will be generated automatically on the save method call, along with some data migrations, management commands, and django admin registrations with custom displays and filters etc.	
	Introduction to REST APIs Django REST Framework Intro Implement token based authentication in DRF	Class Based Views. APIView. Customizing APIView Overriding List View Functions.	Generic Views. Generic APIView Save and Deletion Hooks	Concrete Generic Views and Mixins. Understanding ViewSets.	Writing Unit Tests of the Django App and test coverage integration	TASK: Writing A Django Management Command that will run and store Records in the Database. Inserts Dummy Users in the Database using management commands. It will also store 100 PST date times in a separate table. There will be a scheduled Cron Job, that will run after every 5 minutes and update	

						10 date times in UTC. Once all 100 date times are in UTC, the cron job will start converting date time objects in PST. and this will continue TASK: Writing Authentication APIs, Users List API, User Edit Profile API	
	Add Celery, Celery beat tasks Add Logging	Django Optimizations (Debug Toolbar/Silk, select_related, prefetch_related, values_list, defer, explain) Understanding DB queryplan (Explain) Understanding Indexing Understanding queryset execution, how & when it's executed	API Frameworks (RPC, GraphQL, Schema Definition Language)	RPC Graph QL (queries, mutations, and subscriptions) Graphene & Django Graphene"	Add emails (text and html) with attachment Third party service Integration eg SMS service integration	Task: Add GRAPH QL version based API's alongside REST api's on the existing django project Task: Introduce celery tasks in the existing project TASK: Optimize the existing application	

React	Javascript Variables, Strings, Numbers. (Primitive / non primitives) var, let and const Math Object in JS. (random, rounding values to nearest integers) Functions and Time Out delays. Immuatbility Pass by value and pass by refrences	JS: Objects. (property, methods, prototypes) Understanding **this** keyword. Understanding DOM window, history, document DOM Elements. update innerHTML. DOM maipulation: Selecting Elements in the DOM Traversing the DOM Applying Dynamic Classes. Query params LocalStorage, Session Storage, Cookies	Events and Events Listeners. Event Propagation Event Delegation callbacks and Promises, async await	Package managers: npm, yarn, pnpm Toolchains: Webpack, Vite Transpikers: babel Working with third party scripts (CDN's, via packaage manager) Script tag async vs defer	Introduction to ES6. Learn ES6 concepts like arrow functions, classes, enhanced object literals, template strings, destructuring, default + rest + spread, (for both arrays and objects) let + const, iterators + for..of + enumartors for ... in Named and Default Exports + Classes + What are Pure Functions	TASK: Apply validations (async with the help of promise) on Signup Forms using jS. Add throttle and debounce maintain light dark / theme TASK: Basic setup of toolchains like webpack / vitejs	

	Introduction to React Props Component State Understanding Props vs State Stateless Components Handling Events Life cycle of a component class vs functional components	Controlled Components, Conditional rendering Learn about hooks, Managing state using useState hook	Building List of components using Map. Understanding List Item Keys. Handling Null Props. Debugging react app using React Dev Tools Form fields validation and submission Learn useRef hook to persist values between renders	useEffect vs useLayout Effect Ajax Requests with Axios. UseEffect hook for API calls and handling other sideeffects Learn about Routing for multi-page app, Learn useContext + useReducer hook for sharing data among multiple components	Error boundries Different ways to style: Css in js (emotion etc) css css modules UI libraries (Mui, antd , mantine etc) Lazy loading components Learn about code-splitting for app improving performance Type checking: Prop types, flow and typescript	TASK: Create a point-of-sale app. The site should allow user to add, edit and delete products. User should be able to search products by name. User should be able to upload image of a product (max size 200 x 200) Readable / Appropriate error should be shown on add/edit product form For API, use https://fakestoreapi.com/ TASK: Use Google Youtube Search API and make a React App just like Youtube. It will have a search input and based on the search query, show results. There would be a single selected Item with the Video Player on the left side, just like youtube. Make proper use of CSS to style the app. Feel free to use the boilerplate code if needed.	
	Introduction to Redux (Understanding basics of redux). Modelling Application State.	Managing App State with Redux Toolkit.	Actions, reducers, slices, selectors	Explore middlewares, why there is a need for them Thunk, sagas, redux-observable	Use RTK query to make api calls mutation, caching etc	TASK: Write a React App to show the Temperature, Pressure and Humidity of each City. There will be a search box in which you type a name of a US State and upon pressing Search button, a graph will appear showing Temperature, Pressure and Humidity of that state. Upon searching a new state, it will add the new state at	

						the top and keep the data of previous state there on the page. API: openweathermap.org/forecast5	
	Form Elements in React. React hook form and formik	Navigation on Submit.	Understanding and implementing basic redux middleware			TASK: Create a Blog Application using React. Write Backend in Django Rest Framework, with proper user authentication in place Users should be able to add a new blog, edit a blog, make it public/private, favorite a blog and follow other users. There would be an index page with a list of available blogs (clickable) to redirect on the detail page of the blog.	Exact project idea and requirements to be determined by the trainer, which should cover all the basic Python/Django and ReactJS ideas