Line-by-Line Explanation of POST, GET, PUT, DELETE APIs

1. POST API - postCreateBooking()

```
async postCreateBooking(overrides = {}, statuscode) {

    const dynamicPayload = await this.api.buildPayload(
        payloads.postbooking.booking,
        overrides
    );

    const response = await this.api.request({
        method: "POST",
        url: endpoints.url,
        endpoint: endpoints.getbookingid,
        body: dynamicPayload,
        expectedStatus: statuscode
    });

    this.world.bookingId = response.body.bookingid;

    return response.body;
}
```

Line 1: async makes the function asynchronous and allows use of await. overrides = {} sets default empty object if no overrides provided. statuscode represents expected HTTP status. buildPayload merges base payload with dynamic override values. await pauses execution until payload is ready. this.api.request() calls centralized HTTP engine. method defines HTTP method. url is base URL from config. endpoint is specific API path. body contains JSON payload. expectedStatus validates response status. this.world.bookingId stores created ID in Cucumber World object. return response.body sends only response data back to step definition.

2. GET API - getBookingDetails()

```
async getBookingDetails(enterBookingID = null, statuscode = 200) {

    const bookingId = enterBookingID || this.world?.bookingId;

    if (!bookingId) {
        throw new Error("Booking ID not available");
    }

    const response = await this.api.request({
        method: "GET",
        url: endpoints.url,
        endpoint: endpoints.getbookingdetils + bookingId,
        expectedStatus: statuscode
    });

    return response.body;
}
```

async defines asynchronous function. enterBookingID allows optional external ID. statuscode default is 200. Logical OR chooses passed ID or stored ID from World object. Optional chaining prevents crash if world undefined. If no bookingId exists, error is thrown (defensive programming). this.api.request sends GET request. endpoint dynamically appends bookingId to URL. expectedStatus ensures correct response. return response.body returns JSON response.

3. PUT API - updateBookingDetailsByID()

```
async updateBookingDetailsByID(overrides = {}, statuscode = 200) {

    const bookingId = this.world?.bookingId;

    if (!bookingId) {
        throw new Error("Booking ID not available for update");
    }

    const token = await this.api.tokengenerator();

    const dynamicPayload = await this.api.buildPayload(
        payloads.putbooking.updatebooking,
        overrides
    );

    const response = await this.api.request({
        method: "PUT",
        url: endpoints.url,
        endpoint: endpoints.getbookingdetils + bookingId,
        body: dynamicPayload,
        token: token,
        expectedStatus: statuscode
    });

    return response.body;
}
```

async enables asynchronous execution. bookingId fetched from World object. If bookingId missing, error is thrown. tokengenerator() authenticates and retrieves token. buildPayload creates updated JSON body. method PUT indicates update operation. token is passed for authentication (sent as Cookie). endpoint dynamically includes bookingId. return response.body returns updated booking data.

4. DELETE API - deleteBookingDetailsByID()

```
async deleteBookingDetailsByID(statuscode = 201) {

    const bookingId = this.world?.bookingId;

    if (!bookingId) {
        throw new Error("Booking ID not available for delete");
    }

    const token = await this.api.tokengenerator();

    const response = await this.api.request({
        method: "DELETE",
        url: endpoints.url,
        endpoint: endpoints.getbookingdetils + bookingId,
        token: token,
        expectedStatus: statuscode
    });

    this.world.bookingId = null;

    return response.body;
}
```

async defines asynchronous delete function. statuscode default 201 (resource deleted). bookingId retrieved from World object. If not available, error thrown. tokengenerator() retrieves authentication token. method DELETE removes resource. endpoint dynamically formed using bookingId. token sent for secure request. expectedStatus validates response. this.world.bookingId = null clears scenario state. return response.body returns delete response.