```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import mealAllowanceService from "services/MealAllowance/MealAllowanceService";
import { cloneDeep } from "lodash";

export const getMealAllowances = createAsyncThunk(
  "mealAllowances/getMealAllowances",
  async (data, { rejectWithValue }) => {
    try {
      const response = await mealAllowanceService.Get(data);
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateMealAllowance = createAsyncThunk(
  "mealAllowances/updateMealAllowance",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await mealAllowanceService.Update(data);
      if (onSuccess) onSuccess(response);
      dispatch(getMAById({ id: response.data }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const createMealAllowance = createAsyncThunk(
  "mealAllowances/createMealAllowance",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await mealAllowanceService.Create(data);
      if (onSuccess) onSuccess(response.data);
      dispatch(getMAById({ id: response.data?.id }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);

export const deleteMealAllowance = createAsyncThunk(
  "mealAllowances/deleteMealAllowance",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await mealAllowanceService.Delete(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const getMAById = createAsyncThunk(
  "mealAllowances/getOTById",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
```

```
    try {
      const response = await mealAllowanceService.GetById(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const createMealAllowanceDetail = createAsyncThunk(
  "mealAllowances/createMealAllowanceDetail",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await mealAllowanceService.CreateDetail(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);
export const getMealAllowanceDetail = createAsyncThunk(
  "mealAllowances/getMealAllowanceDetail",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess, id, searchText } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await mealAllowanceService.GetDetail(id, searchText);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);
export const updateMealAllowanceDetail = createAsyncThunk(
  "mealAllowances/updateMealAllowanceDetail",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await mealAllowanceService.UpdateDetail(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);

export const deleteMealAllowanceDetail = createAsyncThunk(
  "mealAllowances/deleteMealAllowanceDetail",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await mealAllowanceService.DeleteDetail(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
```

```
export const updateMATotalEmployee = createAsyncThunk(
  "mealAllowances/updateMATotalEmployee",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await mealAllowanceService.UpdateMATotalEmployee(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  mealAllowanceList: [],
  mealAllowanceDetail: {
    loading: false,
    data: null,
  },
  initForm: {
    name: null,
    date: null,
    price: null,
  },
  mealAllowanceDetailEmp: {
    loadingEmp: false,
    dataEmp: [],
  },
};

const mealAllowancesSlice = createSlice({
  name: "mealAllowances",
  initialState,
  reducers: {
    updateInitForm: (state, action) => {
      state.initForm = { ...state.initForm, ...action.payload };
    },
    resetInitForm: (state) => {
      state.initForm = initialState.initForm;
    },
    resetMADetail: (state) => {
      state.mealAllowanceDetail = initialState.mealAllowanceDetail;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getMealAllowances.pending, (state) => {
        state.loading = true;
      })
      .addCase(getMealAllowances.fulfilled, (state, action) => {
        state.loading = false;
        state.mealAllowanceList = action.payload;
      })
      .addCase(getMealAllowances.rejected, (state) => {
        state.loading = false;
      })
      .addCase(updateMealAllowance.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateMealAllowance.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(updateMealAllowance.rejected, (state) => {
        state.loading = false;
      })
      .addCase(createMealAllowance.pending, (state) => {
        state.loading = true;
      })
      .addCase(createMealAllowance.fulfilled, (state) => {
        state.loading = false;
      })
```

```
.addCase(createMealAllowance.rejected, (state) => {
  state.loading = false;
})
.addCase(deleteMealAllowance.pending, (state) => {
  state.loading = true;
})
.addCase(deleteMealAllowance.fulfilled, (state) => {
  state.loading = false;
})
.addCase(deleteMealAllowance.rejected, (state) => {
  state.loading = false;
})

.addCase(getMAById.pending, (state) => {
  state.loading = true;
  state.mealAllowanceDetail = {
    ...state.mealAllowanceDetail,
    loading: true,
  };
})
.addCase(getMAById.fulfilled, (state, action) => {
  state.loading = false;
  state.mealAllowanceDetail = { data: action.payload, loading: false };
})
.addCase(getMAById.rejected, (state) => {
  state.loading = false;
  state.mealAllowanceDetail = {
    ...state.mealAllowanceDetail,
    loading: false,
  };
})

.addCase(createMealAllowanceDetail.pending, (state) => {
  state.loading = true;
})
.addCase(createMealAllowanceDetail.fulfilled, (state) => {
  state.loading = false;
})
.addCase(createMealAllowanceDetail.rejected, (state) => {
  state.loading = false;
})

.addCase(getMealAllowanceDetail.pending, (state) => {
  state.loading = true;
  state.mealAllowanceDetailEmp = {
    ...state.mealAllowanceDetailEmp,
    loadingEmp: true,
  };
})
.addCase(getMealAllowanceDetail.fulfilled, (state, action) => {
  state.loading = false;
  state.mealAllowanceDetailEmp = {
    loadingEmp: false,
    dataEmp: action.payload,
  };
})
.addCase(getMealAllowanceDetail.rejected, (state) => {
  state.loading = false;
  state.mealAllowanceDetailEmp = {
    ...state.mealAllowanceDetailEmp,
    loadingEmp: false,
  };
})

.addCase(updateMealAllowanceDetail.pending, (state) => {
  state.loading = true;
})
.addCase(updateMealAllowanceDetail.fulfilled, (state) => {
  state.loading = false;
})
.addCase(updateMealAllowanceDetail.rejected, (state) => {
  state.loading = false;
})
```

```
      .addCase(deleteMealAllowanceDetail.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteMealAllowanceDetail.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(deleteMealAllowanceDetail.rejected, (state) => {
        state.loading = false;
      })
      .addCase(updateMATotalEmployee.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateMATotalEmployee.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(updateMATotalEmployee.rejected, (state) => {
        state.loading = false;
      });
  },
});
export const { updateInitForm, resetInitForm, resetMADetail } =
  mealAllowancesSlice.actions;

export default mealAllowancesSlice.reducer;
```