Team Manager:

- Omri Shmuel

Team Members:

Full Name	Id	Phone	Email
1. Omri Shmuel	318688652	050-2578783	Omri164@outlook.com
2. Astewuol Tsagaow	321072670	053-2776230	omerzadi2014@gmail.com
3. Omer Tzaadi	207450628	054-8815358	astul1996@gmail.com

video

App.js

```
const createError = require('http-errors');
const mongoose = require('mongoose');
mongoose.Promise = global.Promise;
const express = require('express');
const path = require('path');
const cookieParser = require('cookie-parser');
const logger = require('morgan');

const indexRouter = require('./routes/index');
const usersRouter = require('./routes/users');
const apiRouter = require('./routes/api');

const connectDB =require('./models/database');

const app = express();

// connect to mongodb
connectDB()
//mongoose.Promise = global.Promise;

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/api', apiRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

api.js

```
const express = require('express');
const router = express.Router();
const Cost = require('../models/costs');
const User = require('../models/users');

/**
 * @route POST /api/add
 * @desc Adds a new cost entry for a user
 * @access Public
 */
router.post('/add', async (req, res) => {
    try {
        const { userid, category, description, sum, date } = req.body;

        // Check if all required fields are provided
        if (!userid || !category || !description || !sum) {
            return res.status(400).json({ error: 'Missing required fields: userid, category, description, sum'
        }

        // If date is not provided, use the current date and time
        const costDate = date ? new Date(date) : new Date();

        // Create a new cost item
        const cost = new Cost({
            userid,
            category,
            description,
            sum,
            date: costDate
        });

        // Save the new cost item to the database
        const savedCost = await cost.save();

        //update total cost for the compute desine
        await User.findOneAndUpdate(
            {id:userid},
            {$inc:{total_cost:sum}},
            {new: true},

        )
        // Respond with the newly added cost item
        res.status(201).json(savedCost);
    } catch (error) {
        // Catch any errors and return them
        res.status(500).json({ error: error.message });
    }
});

 * @route GET /api/report
 * @desc Fetches monthly report for a specific user
 * @access Public
 */
router.get('/report', async (req, res) => {
    const { id: userid, year, month } = req.query;

    // Validate input parameters
    if (!userid || !year || !month) {
        return res.status(400).json({ error: 'Missing required query parameters: id, year, month' });
    }
```

api.jsa

```
    try {
        const startOfMonth = new Date(year, month - 1, 1); // Start of month
        const endOfMonth = new Date(year, month, 0);       // End of month

        // Fetch the costs within the month and year range
        const costs = await Cost.find({
            userid: userid,
            date: { $gte: startOfMonth, $lte: endOfMonth }
        });

        if (costs.length === 0) {
            return res.status(404).json({ error: 'No costs found for this user in the specified month and year'
        }

        // Prepare the response structure
        const categories = {
            food: [],
            education: [],
            health: [],
            housing: []
        };

        // Group costs by category
        costs.forEach(cost => {
            const { category, sum, description, date } = cost;
            const day = new Date(date).getDate(); // Extract the day

            if (categories[category]) {
                categories[category].push({ sum, description, day });
            }
        });

        // Format the final report
        const report = {
            userid: userid,
            year: parseInt(year),
            month: parseInt(month),
            costs: Object.keys(categories).map(category => ({ [category]: categories[category] }))
        };

        res.json(report);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

/**
 * @route GET /api/users/:id
 * @desc Retrieves user details and their total spending
 * @access Public
 */
router.get('/users/:id', async (req, res) => {
    const userId = req.params.id;

    try {
        const user = await User.findOne({ id: userId });
        if (!user) {
            return res.status(404).json({ error: 'User not found' });
        }
```

api.js

```
        res.json({
            first_name: user.first_name,
            last_name: user.last_name,
            id: user.id,
            marital_status: user.marital_status,
            total: user.total_cost,
        });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

/**
 * @route GET /api/about
 * @desc Provides information about the development team
 * @access Public
 */
router.get('/about', async (req, res) => {
    const team = [
        { first_name: 'Omri', last_name: 'Shmuel'},
        { first_name: 'Omer', last_name: 'Tzaadi' },
        { first_name: 'Astewuol', last_name: 'Tsagaow' },
    ];

    res.json(team);
});

module.exports = router;
```

---

database.js

```
const mongoose = require('mongoose');

/**
 * @desciption MongoDB connection utility
 * @modlue database/connection
 */

const uri = "mongodb+srv://OmriShmuel:Omr!3147@cluster0.mrr1c.mongodb.net/CostManager?retryWrites=true&w=majori

/**
 * @func connectDB
 * @desc Attemps to connect to MongoDB using Mongoose
 */
const connectDB = async () => {
    try {
        await mongoose.connect(uri, {
            useNewUrlParser: true,
            useUnifiedTopology: true,
        });
        console.log('MongoDB conected');
    } catch (error) {
        console.error('Databse conection error:', error);
        process.exit(1); // Exit process with failure
    }
};

module.exports = connectDB;
```

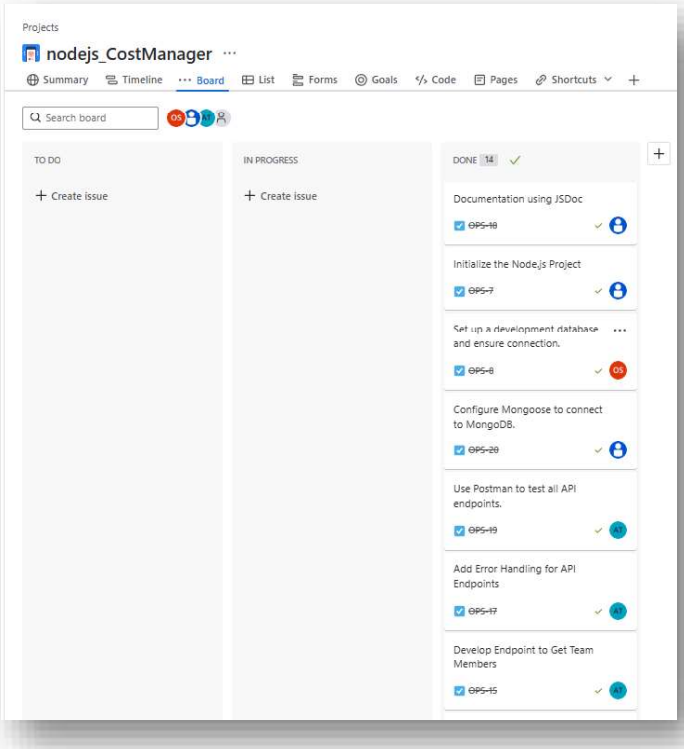
costSchema

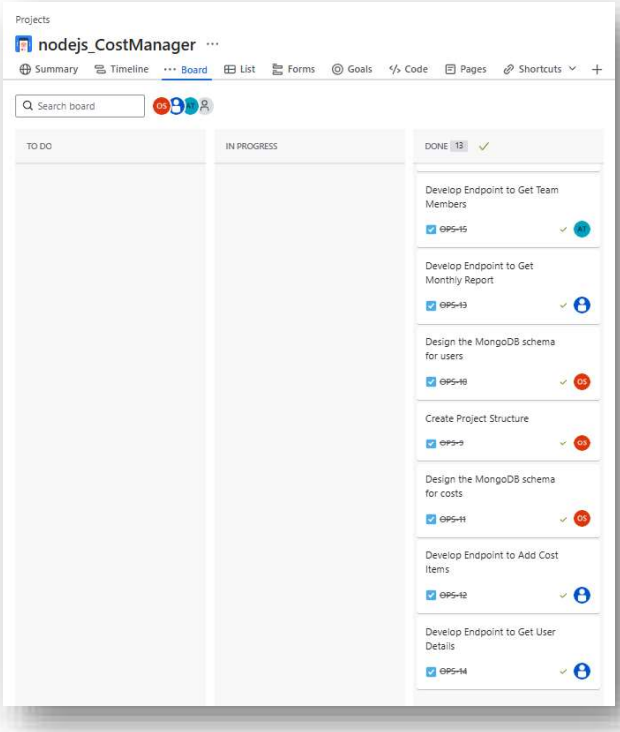
```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

/**
 * @desciption Defines the User schema for MongoDB
 * @modlue models/users
 */

const userSchema = new Schema({
    id: { type: String, required: true }, // Unique user iD
    first_name: { type: String, required: true }, // User's frist name
    last_name: { type: String, required: true }, // User's last name
    birthday: { type: Date, required: true }, // Date of birht
    marital_status: { type: String, required: true }, // Marrital status (single, maried, etc.)
    total_cost: { type: Number, default: 0 }
});

/**
 * @desciption Mongoose model for the Users colecttion
 */
const Users = mongoose.model('users', userSchema);

module.exports = Users;
```

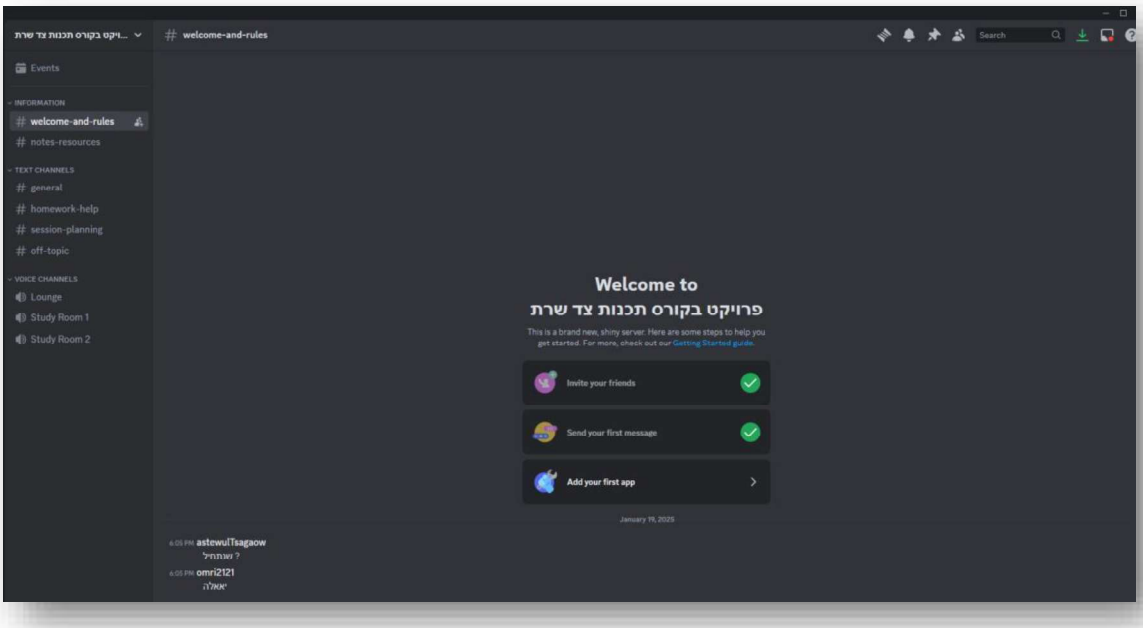
userSchema

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

/**
 * @desciption Defines the User schema for MongoDB
 * @modlue models/users
 */

const userSchema = new Schema({
    id: { type: String, required: true }, // Unique user iD
    first_name: { type: String, required: true }, // User's frist name
    last_name: { type: String, required: true }, // User's last name
    birthday: { type: Date, required: true }, // Date of birht
    marital_status: { type: String, required: true } // Marrital status (single, maried, etc.)
});

/**
 * @desciption Mongoose model for the Users colecttion
 */
const Users = mongoose.model('users', userSchema);

module.exports = Users;
```

Snapshots of Jira, and Slack Alternatives (Discord)






welcome-and-rules
Search
Events
INFORMATION
welcome-and-rules
notes-resources
TEXT CHANNELS
general
homework-help
session-planning
off-topic
VOICE CHANNELS
Lounge
Study Room 1
Study Room 2
Welcome to
פרויקט בקורס תכנות צד שרת
This is a brand new, shiny server. Here are some steps to help you get started. For more, check out our Getting Started guide.
Invite your friends
Send your first message
Add your first app
January 19, 2025
6:05 PM astewulTsagaow
?שתתחיל
6:05 PM omri2121
יאללה